Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Jong In Lim   Dong Hoon Lee (Eds.)

Information Security and Cryptology – ICISC 2003

6th International Conference
Seoul, Korea, November 27-28, 2003
Revised Papers

Springer

Volume Editors

Jong In Lim
Dong Hoon Lee
Korea University
1,5-Ka, Anam-dong Sungbuk-ku, Seoul, 136-701,Korea
E-mail:{jilim/donghlee}@korea.ac.kr

Library of Congress Control Number: 2004102811

CR Subject Classification (1998): E.3, G.2.1, D.4.6, K.6.5, F.2.1, C.2, J.1

ISSN 0302-9743
ISBN 3-540-21376-7 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN: 10992845 06/3142 5 4 3 2 1 0

Binary Tree Encryption: Constructions and Applications

Jonathan Katz[⋆]

Department of Computer Science
University of Maryland
College Park, MD, USA
jkatz@cs.umd.edu

Abstract. *Binary tree encryption* (BTE), a relaxation of hierarchical identity-based encryption (HIBE), has recently emerged as a useful and intriguing primitive. On the one hand, the definition of security for BTE is sufficiently "weak" that — in contrast to HIBE — constructions of BTE *in the standard model* are known. On the other hand, BTE is sufficiently powerful that it yields a number of applications which are important from both a theoretical and a practical point of view.

This survey presents the basic definitions of BTE and also highlights some recent applications of BTE to forward-secure encryption, identity-based and hierarchical identity-based encryption, chosen-ciphertext security, and adaptively-secure encryption.

1 Introduction

The notion of identity-based cryptography has long fascinated researchers [23]. Loosely speaking, in such a scheme *any* identity (i.e., bit-string) can serve as a public key. In somewhat more detail, there is a (trusted) private-key generator PKG who generates master system parameters params along with a master secret key sk. For any identity $id \in \{0,1\}^*$ the PKG can use sk to compute a secret key SK_{id} corresponding to this identity. The pair (id, SK_{id}) then functions as a standard public-/private-key pair (with the important distinction that id can be any string!) whose functionality is determined by the underlying identity-based scheme. (The PKG would presumably authenticate the identity of the person claiming "id" before giving them the corresponding secret key SK_{id}. However, this is outside the scope of the present discussion.) An identity-based system is secure (informally) if knowledge of the secret keys corresponding to any arbitrary-size set of identities $\mathcal{I} = \{id_1, \ldots, id_n\}$ does not allow an adversary to "break" the scheme (in the appropriate sense) for any $id' \notin \mathcal{I}$.

Shamir [23] was the first to suggest an implementation of an identity-based signature scheme. Following this, many provably-secure proposals for identity-based signature and identification schemes followed (e.g., [13, 16]); some of these

[⋆] Portions of this work were supported by NSF grant #ANI-0310751.

constructions were recently generalized and expanded upon in [11]. Although these constructions are proven secure in the random oracle model, note that it is also possible to construct identity-based signatures in the standard model based on any "regular" signature scheme (see [11]).

Recently, Boneh and Franklin [5] and Cocks [10] resolved a long-standing open problem by constructing the first identity-based public-key *encryption* schemes. Both of these constructions are proven secure in the random oracle model. Since encryption schemes are the focus of this article (and are more interesting in the sense that they are more difficult to construct), we consider only encryption from now on.

It is natural to extend the notion of identity-based encryption (IBE) to include *hierarchical* identity-based encryption (HIBE). In an HIBE scheme, the PKG (as above) issues secret keys to "first-level" identities $id \in \{0,1\}^*$; furthermore, anyone knowing the secret key SK_{id_1} corresponding to a "first-level" identity id_1 can issue a secret key $SK_{id_1||id_2}$ corresponding to any "second-level" identity $id_1||id_2$ (for arbitrary $id_2 \in \{0,1\}^*$). More generally, let $ID = (id_1|| \cdots ||id_t)$ and let SK_{ID} be the secret key corresponding to this identity. Then for any string $id_{t+1} \in \{0,1\}^*$ and identity $ID' \stackrel{\text{def}}{=} (ID||id_{t+1})$, knowledge of SK_{ID} enables computation of a key $SK_{ID'}$. As before, in all these cases the pair (ID, SK_{ID}) functions as a "standard" public-/private-key pair. The security requirement is modified in the obvious way: now, one requires that knowledge of the secret keys corresponding to any arbitrary-size set of identities $\mathcal{I} = \{ID_1, \ldots, ID_n\}$ should not enable an adversary to "break" the scheme (in some appropriate sense) for any ID' having no ancestors in $\mathcal{I}$, where the *ancestors* of an identity $ID = (id_1|| \cdots ||id_n)$ are all identities of the form $(id_1|| \cdots ||id_i)$ for $i \leq n$.

Horwitz and Lynn [17] were the first to suggest the notion of HIBE, and they also propose a partial solution handling identities of depth two. Gentry and Silverberg [14] were the first to give a complete solution to this problem, and they construct and prove secure a scheme supporting identities of arbitrary (constant) depth. Both of these constructions build on the IBE scheme of Boneh and Franklin [5], and both are proven secure in the random oracle model.

1.1 Binary Tree Encryption

It can be immediately noticed that the identities in a hierarchical identity-based scheme correspond in the natural way to nodes in a tree. Specifically, one may associate the PKG with the root of the tree, the "first-level" identities with the nodes of depth one (i.e., the children of the root), and the identity $ID' = (id_1|| \cdots ||id_{t+1})$ with a node at depth $t+1$ which is the child of a node at depth t which is in turn associated with $ID = (id_1|| \cdots ||id_t)$.

In a scheme as outlined above, the identity hierarchy yields a tree of *unbounded* degree. In contrast, a *binary* tree encryption (BTE) scheme [7] — as the name suggests — considers only an identity hierarchy in the form of a *binary* tree (i.e., a tree in which each node has degree two). Viewing BTE as a conceptual relaxation of HIBE, one obtains a scheme in which the PKG may potentially issue secret keys to (only) two "identities": 0 and 1. In turn, the "identity" 0 (with knowledge of the appropriate secret key SK_0) can potentially issue secret keys for the "identities" 00 and 01; an analogous statement holds for the "identity" 1. More generally (and dispensing with the purely imaginary concept of "identities" here), the secret key SK_w corresponding to the binary string $w \in \{0,1\}^t$ enables derivation of the secret keys SK_{w0} and SK_{w1} corresponding to the strings $w0, w1 \in \{0,1\}^{t+1}$. As in the case of hierarchical identity-based encryption, each pair (w, SK_w) functions as a public-/private-key pair. A definition of security is developed in a way similar (but slightly different) to that discussed above in the context of hierarchical identity-based encryption; a formal definition appears in Section 2.

"Relaxing" the notion of hierarchical identity-based encryption in this way turns out to be an incredibly powerful idea. For one, a BTE scheme supporting trees of arbitrary *polynomial* depth has recently been constructed and proven secure *in the standard model* [7] (recall that in the case of HIBE, only a scheme of *constant* depth with a proof of security in the *random oracle model* [14] is known). The proof relies on a reasonable number-theoretic assumption (namely, the *decisional bilinear Diffie-Hellman assumption*) related[1] to that used by Boneh and Franklin in constructing their ID-based scheme [5]. This construction of a BTE scheme builds on the construction of Gentry and Silverberg with an important "twist": because a *binary* tree is used (and because of a slight relaxation of the security definition), it is possible to replace the random oracle with a $\mathsf{poly}(k)$-wise independent hash function, a description of which is included as part of the master system parameters.

Equally important, the "relaxed" notion of BTE is surprisingly powerful and suffices for a number of applications:

- BTE was used to construct the first *forward-secure encryption scheme* [7] (indeed, the notion of BTE was introduced in the context of research directed toward solving this problem). Note that this is currently the only methodology known for achieving forward-secure encryption.
- BTE implies both identity-based encryption as well as hierarchical identity-based encryption [7], albeit only with respect to a *non-adaptive* definition of security which is weaker than the definition originally proposed [5]. This results in the first constructions of IBE and HIBE schemes which may be proven secure in the standard model.
- Recent work [8] shows that any IBE scheme (even if "only" secure against non-adaptive attacks) can be used to construct a standard public-key encryption scheme secure against adaptive chosen-ciphertext attacks (i.e., a CCA-secure scheme; cf. [3]). Given the result mentioned above, this yields a new construction of a CCA-secure encryption scheme in the standard model. Interestingly, the construction seems not to follow the paradigms underlying all previous constructions of CCA-secure encryption schemes (cf. [12]).

[1] It is also possible to base a BTE scheme on the identical assumption used by Boneh and Franklin (in the standard model) at the expense of a loss in efficiency.

– Finally, it has recently been shown [9] how to construct an adaptively-secure encryption scheme with "short" keys (namely, with keys shorter than the length of all plaintext messages sent — in fact, the length of plaintext to be encrypted may be *a priori* unbounded) based on any forward-secure encryption scheme plus an NIZK proof system.[2] We comment that adaptively-secure encryption with "short" keys is impossible [21] unless some form of key-evolving techniques (such as those used in forward-secure encryption schemes) are used.

It is hoped that the above results represent just the "tip of the iceberg" and that further applications of BTE will be developed.

1.2 Outline

The remainder of the paper is organized as follows. In Section 2, we give a formal definition of binary tree encryption as well as the corresponding definition of security. In Section 3, we state the known results regarding constructions of BTE. The applications of BTE, as highlighted above, are discussed in Section 4. The treatment given here is at a relatively high level; the interested reader is referred to the original papers [7, 8, 9] for additional information.

2 Definitions

Definitions related to identity-based encryption [5] and hierarchical identity-based encryption [14] are given elsewhere; for the purposes of understanding the definition of binary tree encryption, the informal descriptions provided in the Introduction should suffice. We thus begin with a formal definition of *binary tree encryption* (BTE), taken from [7]:

Definition 1. *A (public-key)* binary tree encryption (BTE) scheme *is a 4-tuple of* PPT *algorithms* (Gen, Der, Enc, Dec) *such that:*

– *The* key generation algorithm Gen *takes as input a security parameter* 1^k *and a value* ℓ *for the depth of the tree. It returns a master public key* PK *and an initial (root) secret key* SK_ε*. (We assume that the values of* k *and* ℓ *are implicit in* PK *and all node secret keys.)*
– *The* key derivation algorithm Der *takes as input* PK*, the name of a node* $w \in \{0,1\}^{<\ell}$*, and its secret key* SK_w*. It returns secret keys* SK_{w0}, SK_{w1} *for the two children of* w*.*
– *The* encryption algorithm Enc *takes as input* PK*, the name of a node* $w \in \{0,1\}^{\leq\ell}$*, and a message* M*. It returns a ciphertext* C*.*
– *The* decryption algorithm Dec *takes as input* PK*, the name of a node* $w \in \{0,1\}^{\leq\ell}$*, its secret key* SK_w*, and a ciphertext* C*. It returns a message* M*.*

[2] Interestingly, it is shown in [7] how to construct an NIZK proof system based on the same number-theoretic assumption used for the forward-secure encryption scheme.

For correctness, we require that for any (PK, SK_ε) *output by* $\mathsf{Gen}(1^k, \ell)$*, any node* $w \in \{0,1\}^{\leq \ell}$ *and secret key* SK_w *correctly generated for this node, and any message* M*, we have* $M = \mathsf{Dec}(PK, w, SK_w, \mathsf{Enc}(PK, w, M))$*.*

The security notion for BTE is largely similar to the security notion for HIBE, with the key difference being that the present definition requires the attacker to commit to the node to be attacked (i.e., the "target node") *in advance*, before seeing the public key and before asking any key exposure queries. This type of attack is called a *selective-node (SN) attack*. While the resulting definition is weaker than a definition which allows the adversary to *adaptively* select the target node, we stress again that this "weaker" definition suffices for all the applications mentioned herein. Furthermore, it is (in part) this weakening of the definition which allows for a construction of BTE in the standard model.

Definition 2. *A BTE scheme is* secure against selective-node, chosen-plaintext attacks *(*SN-CPA*) if for all polynomially-bounded functions* $\ell(\cdot)$*, the advantage of any* PPT *adversary* A *in the following game is negligible in the security parameter:*

1. $A(1^k, \ell(k))$ *outputs a name* $w^* \in \{0,1\}^{\leq \ell(k)}$ *of a node.*
2. *Algorithm* $\mathsf{Gen}(1^k, \ell(k))$ *outputs* (PK, SK_ε)*. In addition, algorithm* $\mathsf{Der}(\cdots)$ *is run to generate the secret keys of all the nodes on the path from the root to* w^* *(we denote this path by* P*), and also the secret keys for the two children of* w^* *(if* $|w^*| < \ell$*).*
3. *The adversary is given* PK *and also the secret keys* $\{SK_w\}$ *for all nodes* w *of the following form:*
   - $w = w'\overline{b}$*, where* $w'b$ *is a prefix of* w^* *and* $b \in \{0,1\}$ *(i.e.,* w *is a sibling of some node in* P*);*
   - $w = w^*0$ *or* $w = w^*1$ *(i.e.,* w *is a child of* w^**; this is only when* $|w^*| < \ell$*).*

   (Note that this allows the adversary to compute $SK_{w'}$ *for* any *node* $w' \in \{0,1\}^{\leq \ell(k)}$ *that is not a prefix of* w^**.)*
4. *The adversary generates a request* $\mathsf{challenge}(M_0, M_1)$*. A random bit* b *is selected and the adversary is given* $C^* = \mathsf{Enc}(PK, w^*, M_b)$*.*

At the end of the game the adversary outputs $b' \in \{0,1\}$*; it succeeds if* $b' = b$*. The adversary's* advantage *is the absolute value of the difference between its success probability and* $1/2$*.*

Security against chosen-ciphertext attacks (denoted SN-CCA) is defined as the obvious extension of the above; see [7] for details.

3 Constructions of Secure BTE Schemes

We limit ourselves to listing the known results regarding constructions of secure BTE schemes, and to a tabulation of their complexity (as a function of the tree depth); the reader is referred to [7] for further details. All constructions mentioned below (indeed, all known constructions of BTE) rely on variants of the so-called *Bilinear Diffie-Hellman (BDH) assumption*. This assumption was first formally defined by Boneh and Franklin [5], motivated by earlier work of Joux [18] and Joux and Nguyen [19].

One of the main results of [7] is the following:

Theorem 1. *Assuming the decisional BDH assumption, there exists a BTE scheme secure in the sense of* SN-CPA.

It is easy to modify the construction so that it relies only on the (possibly weaker) *computational* BDH assumption (this may be done by using a hard-core predicate of the computational BDH problem, and encrypting bit-by-bit). However, this modification comes at the expense of a significant loss of efficiency.

Two generic techniques for achieving chosen-ciphertext security for an *arbitrary* BTE scheme have been proposed. The first [7] relies on non-malleable non-interactive zero-knowledge (NIZK) proofs, adapting an approach due to Naor and Yung [20] and Sahai [22] in the context of making "standard" public-key encryption schemes secure against chosen-ciphertext attacks. Interestingly, in the process of developing this solution it is also shown how non-malleable NIZK may be based on any publicly-verifiable trapdoor predicate (this notion, introduced by [11, 7], generalizes the notion of trapdoor permutations), and furthermore how the decisional BDH assumption naturally gives rise to such predicates. Putting this together gives the following result:

Theorem 2. *Assuming the decisional BDH assumption, there exists a BTE scheme secure in the sense of* SN-CCA.

Because the above relies on NIZK proofs of generic NP statements, it should properly be regarded as a feasibility result rather than as a method for constructing efficient schemes. Recently [8], a more efficient method for achieving chosen-ciphertext security for an arbitrary BTE scheme was proposed; this method (in particular) avoids any zero-knowledge proofs and instead relies on one-time signature schemes (which may be constructed from any BTE scheme). This gives an alternate proof of the above theorem, via a more practical construction.

The above results all hold in the standard model. If one is willing to assume the random oracle model, improved efficiency can be achieved. For one, it should be clear that any HIBE scheme which is secure for a non-adaptive choice of the target identity is also a BTE scheme; thus, the construction of [14] may be used. One way to view this is as simply replacing the poly(k)-wise independent hash function in the construction of [7] by a random oracle (which, of course, is also a poly(k)-wise independent hash function). This leads to improved efficiency since a poly(k)-wise independent hash function is (relatively) expensive to generate and evaluate — in particular, requiring time $O(\text{poly}(k))$ — while for a random oracle these operations are all assumed to take time $O(1)$. Furthermore, essentially the same scheme (with but one additional call to the random oracle) may be based on the (possibly weaker) computational BDH assumption rather than the decisional BDH assumption. Finally, improved efficiency is also

Table 1. Summary of dependencies on the depth of the tree ℓ

	Standard model	Random oracle model
Master key generation time	$\mathcal{O}(\ell)$	$\mathcal{O}(1)$
Encryption/decryption time	$\widetilde{\mathcal{O}}(\ell)$	$\mathcal{O}(\ell)$
Key derivation time	$\mathcal{O}(\ell)$	$\mathcal{O}(1)$
Ciphertext length	$\mathcal{O}(\ell)$	$\mathcal{O}(\ell)$
Public key size	$\mathcal{O}(\ell)$	$\mathcal{O}(1)$
Secret key size	$\mathcal{O}(\ell)$	$\mathcal{O}(\ell)$

possible for BTE schemes achieving chosen-ciphertext security. See [7] for further details.

For completeness, the asymptotic efficiencies of the two constructions secure in the sense of SN-CPA (i.e., in the standard model and in the random oracle model) are given in Table 1.

4 Applications of BTE

We briefly summarize the known applications of BTE.

4.1 Forward Secure Encryption

Cryptographic computations are often carried out on insecure devices for which the threat of key exposure represents a serious and realistic concern. In an effort to mitigate the damage caused by exposure of secret keys stored on such devices, the paradigm of *forward security* was introduced [1, 4]. In a forward-secure scheme, the secret key is updated at regular periods of time (say, at the end of every day), while the public key remains fixed; such schemes guarantee that exposure of the secret key corresponding to a given time period does not enable an adversary to “break” the scheme (in the appropriate sense) for any *prior* time period.

Although a number of forward-secure signature and identification schemes (beginning with [1, 4]) have been proposed, designing a forward-secure (public-key) *encryption* scheme seemed elusive. As outlined in [7], however, BTE schemes can be used to solve this problem, and to give efficient constructions of forward-secure encryption schemes. The basic idea is as follows: let N be the total number of time periods[3] for which the system will operate. Each of these time periods is associated with a node in a binary tree of depth $\lceil \log N \rceil$ in the following way: the i^{th} time period will be associated with the i^{th} node of the tree according to a pre-order traversal, We denote this node by $\langle i \rangle$.

The secret key at period i will consist of: (1) the secret key for node $\langle i \rangle$ in the underlying BTE scheme; and also (2) the secret keys (in the underlying BTE scheme) for all right-children of the path from the root of the tree to $\langle i \rangle$. To

[3] We assume for simplicity that N is fixed in advance; in fact, an *unbounded* number of time periods can be supported [7].

encrypt a message during period i, a sender simply encrypts it for the node $\langle i \rangle$ (again, using the underlying BTE scheme); note that decryption is possible since the secret key at period i includes, in particular, the secret key for node $\langle i \rangle$. It should be noted also that key updates can be done by erasing the secret key for node $\langle i \rangle$ and using the additional secret keys (i.e., those keys belonging to right-children of the path from the root to $\langle i \rangle$) to derive the necessary keys for the next time period. Details are given in [7], where the following is also proven:

Theorem 3. *(Informal:) Given any BTE scheme secure in the sense of* SN-CPA*, the above construction yields a forward-secure encryption scheme.*

Since the above construction requires a binary tree of depth $\log N$ to support N time periods, the scheme itself has all parameters at most poly-logarithmic in the number of time periods (cf. Table 1). In fact, additional improvements are possible. These improvements and various extensions of the above theorem, omitted here for lack of space, are given in [7].

4.2 (Hierarchical) Identity-Based Encryption

It was noted earlier that any HIBE scheme is also trivially a BTE scheme, without any modification. Interestingly, one can also show that a BTE scheme is powerful enough to construct a full-fledged HIBE scheme [7] (or, as a special case, an identity-based scheme), albeit under a slightly weaker definition which requires a non-adaptive choice of the target identity. We sketch the construction here. An HIBE of depth t is constructed using a BTE of depth $k \cdot t$, where k is the security parameter. Identities are hashed to strings of length at most $k \cdot t$ by applying a collision-resistant hash function H to the identities at each level; thus, the identity $(id_1 || \cdots || id_i)$ is mapped to the string $H(id_1) || \cdots || H(id_i)$. It is not hard to show that this gives a secure HIBE, under the relaxed definition of security given above. In fact, because the target identity must be chosen in advance, it is enough for H to be a universal one-way hash function (whose existence is implied by any BTE scheme); thus, we have:

Theorem 4. *Assuming the existence of a BTE scheme secure in the sense of* SN-CPA*, there exists an HIBE scheme of arbitrary polynomial depth secure under a non-adaptive choice of target identity.*

4.3 Chosen-Ciphertext Security

Recently, an interesting connection between identity-based encryption and security against chosen-ciphertext attacks (for "standard" public-key encryption schemes) has been shown [8]. In particular, it was shown how any identity-based encryption scheme can be used to give a simple and efficient construction of a regular encryption scheme secure against chosen-ciphertext attacks (i.e., a "CCA-secure" encryption scheme). The resulting construction avoids the use of any generic NIZK proofs, and furthermore seems not to follow the paradigm of all previously-known constructions of CCA-secure encryption schemes (cf. [12]).

We describe the construction of a CCA-secure standard encryption scheme now: The public key of the scheme will be the master public key PK of the identity-based scheme, while the secret key SK is the corresponding master secret key. To encrypt a message m, the sender generates verification/signature keys (vk, sk) for any one-time signature scheme, and encrypts m for "identity" vk (using the underlying identity-based scheme). The sender signs the resulting ciphertext C to obtain a signature σ, and sends $\langle vk, C, \sigma \rangle$ to the receiver.

The receiver first verifies that σ is a correct signature on C with respect to vk; if not, the ciphertext is rejected. Otherwise, the receiver uses the master secret key SK to generate a decryption key SK_{vk} corresponding to the "identity" vk. Notice that it can then decrypt C using SK_{vk}.

The following is proven in [8]:

Theorem 5. *(Informal:) Given any identity-based encryption scheme secure under a non-adaptive choice of target identity, the above construction yields a CCA-secure encryption scheme.*

Note that the identity-based scheme need only be secure against a *non-adaptive* choice of target identity; combined with Theorems 1 and 4, this results in a new construction of a CCA-secure encryption scheme in the standard model.

It has already been noted above that a similar technique may be used to construct a BTE scheme secure against chosen-ciphertext attacks, starting with any BTE scheme secure in the sense of SN-CPA. Because this gives a slightly stronger result than that of Theorem 2, we state it here for completeness:

Theorem 6. *Assuming the existence of a BTE scheme secure in the sense of* SN-CPA*, there exists a BTE scheme secure in the sense of* SN-CCA*.*

See [8] for further details.

4.4 Adaptively-Secure Encryption

Standard definitions of secure encryption do not guarantee security in the case of *adaptive* corruptions. In a setting where such adaptive corruptions are possible, the encryption scheme should provide the additional guarantee that the information gathered by an adversary when corrupting parties (and, in particular, learning their secret keys) does not give it any additional advantage toward compromising the security of the uncorrupted parties. Very roughly speaking, this requirement may be captured by the existence of a simulator that can generate "dummy" ciphertexts which it can later "open" (by revealing an appropriate secret key) to any given message. (Of course, this must be done in such a way that the revealed secret key "matches" the fixed public key.) Schemes achieving this notion of security are termed *adaptively secure*. We do not further motivate or describe the definition, but instead refer the reader elsewhere [2, 6, 21, 15, 9] for additional discussion.

Nielsen has shown [21] that non-interactive adaptively-secure encryption (in the standard model) is "impossible" unless the secret key is as long as the length of all plaintext messages that are sent. In particular, this implies that no adaptively-secure scheme supporting an a priori *unbounded* number of messages is possible. This result is in stark contrast to the case for encryption schemes which are not adaptively secure.

It is possible to circumvent Nielsen's impossibility result, however, by considering *key-evolving* cryptosystems; i.e., those in which the secret key evolves over time. This suggests using forward-secure encryption as a building block toward building adaptively-secure schemes. (Note that a forward-secure encryption scheme, by itself, is not necessarily adaptively secure.) In fact, a construction of an adaptively-secure encryption scheme based on any forward-secure encryption scheme has recently been given [9]:

Theorem 7. *(Informal:) Assuming the existence of a forward-secure encryption scheme and an NIZK proof system for NP, there exists a non-interactive adaptively-secure encryption scheme for an unbounded number of messages.*

Since both a forward-secure encryption scheme as well as NIZK proofs may be based on the BDH assumption (cf. [7]), the BDH assumption suffices to imply the result of the theorem.

More efficient constructions of adaptively-secure encryption, which avoid NIZK proofs altogether (but which in some cases require additional number theoretic assumptions), are also given in [9].

Acknowledgements

It was great fun to collaborate with Ran Canetti and Shai Halevi on the all work described herein. I would also like to thank the program chairs for ICISC 2003 (Jong In Lim and Dong Hoon Lee) for inviting me to present this survey, and for a very enjoyable visit to Korea.

References

[1] R. Anderson. Two Remarks on Public Key Cryptology. Invited Lecture, *4th ACM Conference on Computer and Communications Security*, 1997. Available at `http://www.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf`. 7

[2] D. Beaver and S. Haber. Cryptographic Protocols Secure Against Dynamic Adversaries. *Adv. in Cryptology — Eurocrypt 1992*, LNCS vol. 658, Springer-Verlag, pp. 307–323, 1992. 9

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. *Adv. in Cryptology — Crypto 1998*, LNCS vol. 1462, Springer-Verlag, pp. 26–45, 1998. 3

[4] M. Bellare and S. Miner. A Forward-Secure Digital Signature Scheme. *Adv. in Cryptology — Crypto 1997*, LNCS vol. 1666, Springer-Verlag, pp. 75–89, 1997. 7

[5] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *Adv. in Cryptology — Crypto 2001*, LNCS vol. 2139, Springer-Verlag, pp. 213–229, 2001. Full version to appear in *SIAM J. Computing* and available at `http://eprint.iacr.org/2001/090`. 2, 3, 4, 6

[6] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively Secure Multi-Party Computation. *28th ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 639–648, 1996. 9

[7] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. *Adv. in Cryptology — Eurocrypt 2003*, LNCS vol. 2656, Springer-Verlag, pp. 255–271, 2003. Full version available at `http://eprint.iacr.org/2003/083`. 2, 3, 4, 5, 6, 7, 8, 10

[8] R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. Available at `http://eprint.iacr.org/2003/182`. 3, 4, 6, 8, 9

[9] R. Canetti, S. Halevi, and J. Katz. Flash Encryption: Adaptive and Forward Security with Short Keys. Manuscript, November 2003. 4, 9, 10

[10] C. Cocks. An Identity-Based Encryption Scheme Based on Quadratic Residues. *Cryptography and Coding*, LNCS vol. 2260, Springer-Verlag, pp. 360–363, 2001. 2

[11] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong Key-Insulated Signature Schemes. *PKC 2003*, LNCS vol. 2567, pp. 130–144, Springer-Verlag, 2003. 2, 6

[12] E. Elkin and A. Sahai. A Unified Methodology For Constructing Public-Key Encryption Schemes Secure Against Adaptive Chosen-Ciphertext Attack. To appear, *1st Theory of Cryptography Conference*, 2004. Available at `http://eprint.iacr.org/2002/042`. 3, 8

[13] A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. *Adv. in Cryptology — Crypto 1986*, LNCS vol. 263, Springer-Verlag, pp. 186–194, 1986. 1

[14] C. Gentry and A. Silverberg. Hierarchical Identity-Based Cryptography. *Adv. in Cryptology — Asiacrypt 2002*, LNCS vol. 2501, Springer-Verlag, pp. 548–566, 2002. 2, 3, 4, 6

[15] O. Goldreich. Draft of a Chapter on Cryptographic Protocols. Manuscript, June 2003. Available at `http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html` 9

[16] L. Guillou and J.-J. Quisquater. A "Paradoxical" Identity-Based Signature Schemes Resulting from Zero-Knowledge. *Adv. in Cryptology — Crypto 1988*, LNCS vol. 403, Springer-Verlag, pp. 216–231, 1988. 1

[17] J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. *Adv. in Cryptology — Eurocrypt 2002*, LNCS vol. 2332, Springer-Verlag, pp. 466–481, 2002. 2

[18] A. Joux. A One-Round Protocol for Tri-Partite Diffie Hellman. *Fourth Algorithmic Number Theory Symposium (ANTS)*, LNCS vol. 1838, Springer-Verlag, pp. 385–394, 2000. 6

[19] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups. Manuscript, January 2001. Available at `http://eprint.iacr.org/2001/003/`. 6

[20] M. Naor and M. Yung. Public Key Cryptosystems Provably Secure Against Chosen-Ciphertext Attacks. *22nd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 427–437, 1990. 6

[21] J. B. Nielsen. Separating Random Oracle Proofs from Complexity-Theoretic Proofs: The Non-Committing Encryption Case. *Adv. in Cryptology — Crypto 2002*, LNCS vol. 2442, Springer-Verlag, pp. 111–126, 2002. 4, 9

[22] A. Sahai. Non-malleable Non-Interactive Zero-Knowledge and Adaptive Chosen-Ciphertext Security. *40th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 543–553, 1999. 6

[23] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. *Adv. in Cryptology — Crypto 1984*, LNCS vol. 196, Springer-Verlag, pp. 47–53, 1984. 1

A Separable Threshold Ring Signature Scheme

Joseph K. Liu[1], Victor K. Wei[1], and Duncan S. Wong[2]*

[1] Department of Information Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
{ksliu9,kwwei}@ie.cuhk.edu.hk
[2] Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
duncan@cityu.edu.hk

Abstract. We present a threshold ring signature scheme (spontaneous anonymous threshold signature scheme) that allows the use of both RSA-based and DL-based public keys at the same time. More generally, the scheme supports the mixture of public keys for any trapdoor-one-way type as well as three-move type signature schemes. This kind of 'separability' has useful applications in practice as a threshold ring signature is no longer limited to support only one particular type of public keys, as required by all the previous schemes. In the paper, we also show that the signature maintains the anonymity of participating signers unconditionally and is existential unforgeable against chosen message attacks in the random oracle model.

1 Introduction

Let us consider the following scenario. Suppose there are n users in a public-key system in which each of them has a public key pair for digital signature. Their public keys are generated independently without any coordination with others and therefore their keys may correspond to different security parameters. In addition, it is very likely that the keys are for entirely different signature schemes. For example, user 1 may have a key pair for 2048-bit RSA signature scheme [17], user 2 may have one for 1024-bit Digital Signature Algorithm [12], user 3 may have one for 163-bit ECDSA (Elliptic Curve Digital Signature Algorithm) [9], while user 4 may prefer to use one for 1024-bit Schnorr signature scheme [18]. Other users may pick key pairs for their favorable signature schemes other than the previous ones. Under this system setup, t users spontaneously form a group of n possible signers by arbitrarily choose n users including themselves, where $n > t$, and generate a signature for some message such that (1) any public verifier (who has all the n public keys) can tell if the signature is valid (*public verifiability*); (2) can identify who the t actual signers are even when all the

* A part of this work was supported by a grant from CityU(Project No. 7200005).

private keys of the n possible signers in the group are known (*anonymity, signer-ambiguity*); and (3) knowing only $t-1$ private keys of the signing group is not enough for forging such a signature (*unforgeability, robustness*). In this paper, we focus on solving the problem and we call the scheme a *separable threshold ring signature scheme* or *spontaneous anonymous threshold signature scheme*.

The notion of ring signature was first formalized by Rivest et al. in [16]. A ring signature scheme allows a signer to *spontaneously* form a group of signers including himself and generate a signature such that any public verifier can tell if the signature is valid while cannot determine the actual authorship of the signature. In addition, the formation of a signing group does not require any coordination of group membership or any TTP (Trusted Third Party), and therefore can be formed by the actual signer without getting any consent from other diversion group members — *spontaneity*. The concept of spontaneity was absent from most previous results on threshold cryptography and threshold signature schemes [7]. The signing group is formed with the intervention of the group manager and may also require coordination among group members. Also the group manager who always knows who the the actual signer is. In the ring signature, the absence of a group manager provides anonymity to the actual signer both inside and outside the signing group.

A (t,n)-threshold ring signature [3] has the similar notion to the ring signature. First, a (t,n)-threshold ring signature scheme requires at least t signers to work jointly for generating a signature. Second, the anonymity of signers is preserved both inside and outside the signing group. Third, those t participating signers can choose any set of n entities including themselves without getting any consent from those diversion group members.

(Separability) An application is said to be separable if all participants can choose their keys independently with different parameter domains and for different types of signature schemes. The term separability was originated from [10] and was diversified in [5]. There exists weaker forms of separability. Partial separability allows only a subset of the participants to choose their keys independently and weak separability requires all participants to use common system parameters. [**?**] provides perfect separability, [10] provides partial separability and [5] provides weak separability. However, all of these schemes are group signature schemes and there is no corresponding variant of threshold ring signature schemes.

There is another scenario: when all participants in an application choose their keys independently but requiring that all the keys are for one single type of signature schemes. For example, all keys have to be for a RSA-based signature scheme but they may have different key sizes. We say that the application has *type-restricted* separability.

1.1 Our Contributions

We present a (t,n)-threshold ring signature scheme (spontaneous threshold signature scheme) that enjoys separability. In each signature, the public keys indicated can be for any trapdoor-one-way type and any three-move type signature schemes. When compared with [3, 20], our scheme runs more efficiently and offers a smaller signature size in the general case. When compared with [4, 20], our scheme eliminates the need of an ideal cipher and costly domain adjustment. The assumption of the existence of an ideal cipher is stronger than the assumption of having random oracles. Our scheme is proven to be secure in the random oracle model [2].

The remaining of the paper is organized as follows. In Sec. 2, several ring signature schemes and threshold ring signature schemes are reviewed. This is followed by the formal definition of a secure (t, n)-threshold ring signature scheme in Sec. 3. Then we exemplify our scheme using some concrete setup and analyze its security in Sec. 4 and Sec. 5, respectively. In Sec. 6, the generic version of our separable threshold ring signature scheme is presented. We conclude the paper in Sec. 7.

2 Related Work

The notion of ring signature was first formalized by Rivest et al. in [16]. They also proposed an efficient ring signature scheme which support public keys for trapdoor one-way functions. Several other ring signature schemes were later proposed [3, 21, 1, 20]. Among these schemes, [1] supports the use of various flavors of public keys at the same time in one single ring signature. Their scheme allows the mixture of public keys for trapdoor one-way function based signature schemes (also known as *trapdoor-one-way type* signature schemes) and *three-move type* signature schemes such as Fiat-Shamir signature scheme [8] and Schnorr signature scheme [18].

2.1 Witness Indistinguishable Signature (WIS) [6]

Although ring signature was first formalized in 2001 by Rivest, et al. [16], similar concept was actually raised earlier. In 1994, Cramer et al. [6] proposed a proof of knowledge protocol which consists of the properties of a threshold ring signature scheme at large. Their protocol lets a prover show that he knows solutions of at least t out of n problem instances without revealing which t instances are involved. Our (t, n)-threshold ring signature scheme follows its concept in the sense that the t participating signers show that they know at least t private keys corresponding to n public keys without leaking which ones they really are.

The protocol proposed in [6] requires two mechanisms with certain security properties. The first mechanism is a proof of knowledge protocol taken between a prover and a verifier. For simplicity, it is assumed to be a three round public coin protocol where the prover speaks first. The three messages are denoted as m', c and m'' where c is referred to as a challenge generated by the verifier and m'' is the prover's final message which is referred to as the answer. By running this protocol, the prover tries to convince the verifier that he knows the solution (the witness), denoted by w, of a publicly known problem instance, denoted by x. One important property of the proof of knowledge protocol is called honest verifier zero-knowledge. This property allows anyone on input x simulates the conversation (m', c, m'') in the way that it is indistinguishable from a real conversation between an honest prover and an honest verifier.

The second mechanism is a secret sharing scheme. The scheme distributes a secret s among n participants. Each participant has one share. The shares are computed in such a way that knowing t (where $t < n$) shares can reconstruct s. One important property of the scheme is that given s and shares of any $n-t$ participants, the shares of the remaining t participants can be reconstructed.

Let P and V be the prover and verifier, respectively. The description below shows the major steps of the protocol of [6]. It allows P to show V that he knows solutions of t out of n question instances, denoted by $x_1, \cdots, x_n$, without revealing which t instances are involved. Let A be the set of t question instances whose solutions are known to P. Let $\overline{A}$ be the set of the other $n-t$ question instances whose solutions are not known to P.

1. For each $i \in \overline{A}$, as P does not know the solution w_i to question instance x_i, he simulates a conversation (m'_i, c_i, m''_i). For each $i \in A$, P determines m'_i as what an honest prover would send as the first message in the proof of knowledge protocol for the question instance x_i. P sends $m'_1, \cdots, m'_n$ to V.
2. V randomly picks a binary string with appropriate length and denotes it as s, the secret. V sends s to P.
3. Due to the important property of the secret sharing scheme described above, P considers s as the secret and c_i for $i \in \overline{A}$ as the $n-t$ shares, and reconstructs the remaining t shares. These t shares are set to c_i for $i \in A$. In Step 1, m''_i has produced for each $i \in \overline{A}$. For each $i \in A$, P knows the witness w_i for x_i, and can therefore find a valid m''_i for m'_i and c_i by running the proof of knowledge protocol for common input x_i while behaving exactly like an honest prover. P sends the set of messages c_i, m''_i, for $1 \leq i \leq n$, to V.
4. V verifies whether all conversations (m'_i, c_i, m''_i), for $1 \leq i \leq n$, lead to acceptance by honest verifiers in the corresponding proof of knowledge protocols. In addition, V checks if all the n shares are consistent with the secret s. He accepts if all these checkings are satisfied.

It is also remarked in [6] that this protocol can be extended to an non-interactive forum (signature) by computing the challenge as a hash value of the message to be signed and the prover's first message.

In [1], Abe et al. presented an instantiation of [6] which is a ring signature scheme. In [4], Bresson et al. presented a threshold ring signature instantiation of [6]. In [3], Bresson et al. proposed another threshold ring signature scheme with a different construction from [6]. It uses the concept of partitioning. In [20], Wong et al. proposed a threshold ring signature scheme using tandem construction method which can be considered as an extension of the ring signature scheme proposed by Rivest et al. [16]. It runs efficiently when t is near 1 or n. All of them support the use of public keys for only trapdoor one-way type signature schemes. They do not support public keys for three-move type signature schemes or the mixture of these two types of signature schemes. In other words, none of the current threshold ring signature schemes is separable.

In this paper, we propose a separable threshold ring signature scheme. In the following, we first specify the security requirements of a secure threshold ring signature scheme.

3 Security Model

Let $k \in \mathbb{N}$ be a security parameter and $m \in \{0,1\}^*$ be a message.

Definition 1 (Threshold Ring Signature Scheme). *A (t,n)-threshold ring signature scheme is a triple $(\mathcal{G}, \mathcal{S}_{t,n}, \mathcal{V}_{t,n})$ where*

- $(\hat{s}, P) \leftarrow \mathcal{G}(1^k)$ *is a probabilistic polynomial time algorithm (PPT) which takes as input a security parameter k, produces a private key $\hat{s}$ and a public key P.*
- $\sigma \leftarrow \mathcal{S}_{t,n}(1^k, \hat{S}, L, m)$ *is a PPT which accepts as inputs a security parameter k, a set of private keys $\hat{S}$, a set of public keys L including the ones that correspond to the private keys in $\hat{S}$ and a message m, produces a signature σ. We require that $|\hat{S}| = t$ and $|L| = n$ where $0 < t \le n$.*
- $1/0 \leftarrow \mathcal{V}_{t,n}(1^k, L, m, \sigma)$ *is a polynomial-time algorithm which accepts as inputs a security parameter k, a set of n public keys L, a message m and a signature σ, returns 1 or 0 for* accept *or* reject*, respectively. We require that $\mathcal{V}_{t,n}(1^k, L, m, \mathcal{S}_{t,n}(1^k, \hat{S}, L, m)) = 1$ for any message m and any set $\hat{S}$ of t private keys in which each key is generated by $\mathcal{G}(1^k)$ and any set L of n public keys in which the public keys corresponding to all the private keys of $\hat{S}$ are included.*

For simplicity, we usually omit the input of security parameter when using $\mathcal{S}_{t,n}$ and $\mathcal{V}_{t,n}$ in the rest of the paper.

L may include public keys based on different security parameters. The security of the signature scheme defined above is set to the smallest one among them. $\mathcal{G}$ may also be extended to take the description of key types.

The security of a (t,n)-threshold ring signature scheme consists of two requirements, namely *Signer Ambiguity* and *Unforgeability*. They are defined as follows.

Definition 2 (Signer Ambiguity). *Let $L = \{P_1, \cdots, P_n\}$ where each key is generated as $(\hat{s}_i, P_i) \leftarrow \mathcal{G}(1^{k_i})$ for some $k_i \in \mathbb{N}$. Let $k = \min(k_1, \cdots, k_n)$. A (t,n)-threshold ring signature scheme is unconditionally signer ambiguous if, for any L, any message m, and any signature $\sigma \leftarrow \mathcal{S}_{t,n}(\hat{S}, L, m)$ where $\hat{S} \subseteq \{\hat{s}_1, \cdots, \hat{s}_n\}$ and $|\hat{S}| = t$, any unbound adversary E accepts as inputs L, m and σ, outputs π such that $\hat{s}_\pi \in \hat{S}$ with probability t/n.*

It means that even all the private keys are known, it remains uncertain that which t signers out of n possible signers actually work jointly to generate a (t,n)-threshold ring signature.

Definition 3 (Unforgeability). *Let $L = \{P_1, \cdots, P_n\}$ be the set of n public keys in which each key is generated as $(\hat{s}_i, P_i) \leftarrow \mathcal{G}(1^{k_i})$ where $k_i \in \mathbb{N}$. Let $k = \min(k_1, \cdots, k_n)$. Let $\mathcal{SO}(L', i_1, \cdots, i_{t'}, m)$ be a signing oracle that takes any set L' of public keys, where $L' \subseteq L$ and $n' = |L'|$, any t' signers indexed by $i_1, \cdots, i_{t'}$ where $1 \le i_j \le n$, $1 \le j \le t'$ and $t' \le n'$, and any message m, produces a (t', n')-threshold ring signature $\sigma \leftarrow \mathcal{S}_{t',n'}(\{\hat{s}_{i_1}, \cdots, \hat{s}_{i_{t'}}\}, L', m)$ such that $\mathcal{V}_{t',n'}(L', m, \sigma) = 1$. Let $\hat{S}_{t-1}$ be any set of $t-1$ private keys corresponding to the public keys in L. A (t, n)-threshold ring signature scheme is unforgeable if, for any PPT $\mathcal{A}$ with signing oracle $\mathcal{SO}$, for any L, and for all sufficiently large k,*

$$\Pr[\mathcal{A}^{\mathcal{SO}}(1^k, L, \hat{S}_{t-1}) \rightarrow (m, \sigma) : 1 \leftarrow \mathcal{V}_{t,n}(L, m, \sigma)] \le \epsilon(k)$$

where ϵ is some negligible function. Restriction is that $(L, i_1, \cdots, i_t, m, \sigma)$ should not be found in the set of all oracle queries and replies between $\mathcal{A}$ and $\mathcal{SO}$ for any $1 \le i_j \le n$, $1 \le j \le t$. The probability is taken over all the possible inputs of $\mathcal{A}$, oracle queries and coin flips of $\mathcal{A}$.

A real-valued function ϵ is *negligible* if for every $c > 0$, there exists a $k_c > 0$ such that $\epsilon(k) < k^{-c}$ for all $k > k_c$.

We say that a (t, n)-threshold ring signature scheme is *secure* if it satisfies the above requirements.

4 The Separable Threshold Ring Signature Scheme

Our scheme follows the concept of the three-round protocol in [6] (Sec. 2.1). In the following, we exemplify our separable threshold signature scheme by using a concrete setup which consists of the mixture of public keys for all RSA-based and DL-based signature schemes. In Sec. 6, we generalize it to support the mixture of public keys for any trapdoor one-way type and three-move type signature schemes.

4.1 Concrete Setup: All RSA-based and DL-based Public Keys

For $i = 1, \cdots, n$, if user i has a DL-based key pair, his public key is (p_i, q_i, g_i, y_i) and private key is $x_i \in \mathbb{Z}_{q_i}$ where p_i, q_i are prime, $q_i | (p_i - 1)$, $g_i \in \mathbb{Z}_{p_i}^*$ of order q_i and $y_i = g_i^{x_i} \bmod p_i$. We assume that the discrete logarithm problem modulo p_i is hard. If user i has a RSA-based key pair, then his public key is (e_i, N_i) and private key is d_i where N_i is a product of two equal-length prime numbers and $e_i d_i \equiv 1 \pmod{\phi(N_i)}$. We assume that the underlying RSA inverting problem is hard. Let $H_i : \{0,1\}^* \rightarrow \mathbb{Z}_{N_i}$ be some cryptographic hash function. Let L be the set of all public keys of the n users.

Let ρ be twice the bit length of the largest q_i and N_i, for $1 \le i \le n$. Let $G : \{0,1\}^* \rightarrow \{0,1\}^\rho$ be some cryptographic hash function. Without loss of generality, suppose user j, for $1 \le j \le t$, are participating signers and user i, for $t + 1 \le i \le n$, are non-participating signers. To generate a (t, n)-threshold ring signature on a message $m \in \{0,1\}^*$, the t participating signers carry out the following steps.

The Signing Algorithm.

1. For $i = t+1, \cdots, n$, pick $c_i \in_R \{0,1\}^\rho$ and $s_i \in_R \mathbb{Z}_X$ where $X = q_i$ or $X = N_i$ if user i has a DL-based key pair or a RSA-based key pair, respectively. Compute
$$z_i = \begin{cases} g_i^{s_i} y_i^{c_i} \bmod p_i & \text{if user } i \text{ has a DL-based key pair} \\ H_i(c_i) + s_i^{e_i} \bmod N_i & \text{if user } i \text{ has a RSA-based key pair} \end{cases}$$
2. For $j = 1, \cdots, t$, pick $r_j \in_R \mathbb{Z}_X$ and compute
$$z_j = \begin{cases} g_j^{r_j} \bmod p_i & \text{if user } j \text{ has a DL-based key pair} \\ r_j & \text{if user } j \text{ has a RSA-based key pair} \end{cases}$$
3. Compute $c_0 = G(L, t, m, z_1, \cdots, z_n)$ and construct a polynomial f over $GF(2^\rho)$ such that $\deg(f) = n - t$, $f(0) = c_0$ and $f(i) = c_i$, for $t+1 \le i \le n$.
4. For $j = 1, \cdots, t$, compute $c_j = f(j)$ and
$$s_j = \begin{cases} r_j - c_j x_j \bmod q_j & \text{if user } j \text{ has a DL-based key pair} \\ (r_j - H_j(c_j))^{d_j} \bmod N_j & \text{if user } j \text{ has a RSA-based key pair} \end{cases}$$
5. Output the signature for m and L as $\sigma = (s_1, \cdots, s_n, f)$.

The Verification Algorithm. A verifier checks a signature $\sigma = (s_1, \cdots, s_n, f)$ with a message m and a set of public keys L as follows.

1. Check if $\deg(f) = n-t$. If yes, proceed. Otherwise, reject.
2. For $i = 1, \cdots, n$, compute $c_i = f(i)$ and
$$z_i' = \begin{cases} g_i^{s_i} y_i^{c_i} \bmod p_i & \text{if user } i \text{ has a DL-based key pair} \\ H_i(c_i) + s_i^{e_i} \bmod N_i & \text{if user } i \text{ has a RSA-based key pair} \end{cases}$$
3. Check whether $f(0) \overset{?}{=} G(L, t, m, z_1', \cdots, z_n')$. If yes, accept. Otherwise, reject.

It is not difficult to see that this concrete setup can be generalized easily to support the mixture of public keys for all trapdoor-one-way type and three-move type signature schemes at the same time in a signature. A generic scheme is given in Sec. 6.

5 Security Analysis

We first define the following two problems and assume that they are hard in our concrete setup described above.

Discrete Logarithm Problem (DLP). Let p and q be prime, $q|(p-1)$, and $|q| = k$ where k is a security parameter. Let $g \in \mathbb{Z}_p^*$ of order q. Given an element $h \in \mathbb{Z}_p^*$ chosen at random, find an integer x such that $g^x \equiv h \pmod p$.

RSA Problem (RSAP). Let $n = pq$ where p and q are two equal-length primes. Let e be a number such that e and $\phi(n)$ are co-prime. Given an element $y \in \mathbb{Z}_n$ chosen at random, find an integer x such that $x^e \equiv y \pmod{n}$.

Theorem 1. *The scheme in Sec. 4.1 is unconditionally signer ambiguous if all hash functions behave like random oracles [2].*

Proof. The polynomial f, with degree $n{-}t$, is determined by $c_{t+1}, \cdots, c_n$ and c_0. $c_{t+1}, \cdots, c_n$ are randomly generated and c_0 is the output of the random oracle G. Thus f can be considered as a function chosen randomly from the collection of all polynomials over $GF(2^\rho)$ with degree $n{-}t$. Then the distributions of $c_1, \cdots, c_t$ are also uniform over the underlying range.

For $i = t+1, \cdots, n$, s_i are chosen independently and distributed uniformly over $\mathbb{Z}_X$ where $X = q_i$ or $X = N_i$ if signer i has a DL-based key pair or a RSA-based key pair, respectively. For $j = 1, \cdots, t$, r_j are chosen independently and distributed uniformly over $\mathbb{Z}_X$. Since r_j are independent of c_j and the private keys x_j (DL-based private key) or d_j (RSA-based private key), s_j, $1 \le j \le t$, are also uniformly distributed.

In addition, for any fixed message m and fixed set of public keys L, we can see that $(s_1, \cdots, s_n)$ has exactly

$$\prod_{1 \le i \le n} X_i$$

possible solutions. Since the distribution of these possible solutions are independent and uniformly distributed no matter which t participating signers are, an adversary, even has all the private keys and unbound computing resources, has no advantage in identifying any one of the participating signers over random guessing. □

In the following, we show the unforgeability (Def. 3) in the case when all public keys are for DL-based signature schemes.

Lemma 1. *Let a forger E be a probabilistic polynomial time machine. For some message $m \in \{0,1\}^*$ and a set of n public keys L corresponding to n signers, suppose E on inputs the private keys of any $t-1$ signers among the n signers, queries a signing oracle $\mathcal{SO}$ (specified in Def. 3) and a random oracle G polynomial times, and outputs a forged signature σ (i.e. $1 \leftarrow \mathcal{V}_{t,n}(L, m, \sigma)$), with non-negligible probability p_E. Then DLP can be solved with non-negligible probability in polynomial time.*

Proof in Appendix A.

We will need the following definitions.

The RSA-Collision Problem: Given an RSA public key (n, e) and a hashing function H, produce $(c_1, s_1) \neq (c_2, s_2)$ satisfying $H(c_1) + s_1^e = H(c_2) + s_2^e \bmod n$.

Remark: Under the random oracle model, the RSA-Collision Problem is equivalent to the Additive RSA Problem from [10]: Given an RSA public key

(n, e) and a random number r, produce (s_1, s_2) satisfying $s_1^e - s_2^e = r \bmod n$. The proof of this equivalence is straightforward and omitted.

Now we show the unforgeability (Def. 3) in the case that all public keys are for RSA-based signature schemes.

Lemma 2. *Let a forger $\mathcal{A}$ be a probabilistic polynomial time machine. For some message m and a set of n public keys L corresponding to n signers, suppose $\mathcal{A}$ on inputs the security parameter k, the private keys of any $t-1$ signers among the n signers, queries a signing oracle $\mathcal{SO}$ (specified in Def. 3) for q_S times, random oracle G for q_G times and random oracles $\{H_i\}_{1 \leq i \leq n}$ for q_H times combined, and outputs a forged signature σ (i.e. $1 \leftarrow \mathcal{V}_{t,n}(L, m, \sigma)$), with non-negligible probability $p_{\mathcal{A}}$. Then RSA-Collision can be solved with non-negligible probability in polynomial time.*

Proof in Appendix B.

Combining the two lemmas above and some additional derivations, we show that our scheme in the case of having the mixture of keys for RSA-based and DL-based signature schemes is unforgeable.

Theorem 2. *For a PPT forger $\mathcal{F}$ which simulates both forgers specified in Lemma 1 and Lemma 2, suppose $\mathcal{F}$ on inputs the security parameter k, the private keys of any $t-1$ signers among n signers, queries signing oracle $\mathcal{SO}$ for q_S times, random oracles H_i, $1 \leq i \leq n$ for q_H times and G for q_G times, and outputs a forged signature σ with non-negligible probability $p_{\mathcal{F}}$. Then a PPT D can be constructed from $\mathcal{F}$ that solves DLP or RSA-Collision with non-negligible probability.*

Proof. The proof is straightforward by combining the machines B and $\mathcal{M}$ together and replacing the key pairs of other signers to the mixture of RSA-based and DL-based ones. □

6 The Generic Separable Threshold Ring Signature Scheme

In Sec. 4, we exemplify our scheme with support to the mixture of public keys for all types of RSA-based and DL-based signature schemes in a signature. In this section, we generalize it by presenting our Generic Scheme. The scheme can produce a signature which includes the mixture of public keys for any trapdoor-one-way type and three-move type signature schemes. In the following, we first review these two types of signature schemes due to [1].

Trapdoor-One-Way Type Signature Scheme. Let F be a trapdoor one-way function family indexed by a public key P and F^{-1} be the family of its inverse functions indexed by the corresponding private key $\hat{s}$. Let Domain be a function which accepts as input a function and returns the domain of the function. We require that for any $m \in \mathsf{Domain}(F)$, $m = F^{-1}(\hat{s}, F(P, m))$. Assume that computing F^{-1} is hard without knowing $\hat{s}$. For short, we define $F_P(\cdot) = F(P, \cdot)$ and $F_{\hat{s}}^{-1}(\cdot) = F^{-1}(\hat{s}, \cdot)$. Examples of trapdoor one-way type signature schemes include RSA [17] signature scheme, Rabin's function [15, 19], etc. A signing algorithm $\mathcal{S}^{\mathsf{H}}$, which takes a message m and the private key $\hat{s}$, produces a signature σ by

$$\sigma = F_{\hat{s}}^{-1}(H(m))$$

where $H : \{0,1\}^* \to \Delta$ is some appropriate cryptographic hash function and Δ is the range of F. The verification algorithm $\mathcal{V}^{\mathsf{H}}$, which takes a message m, the public key P, and a signature σ, produces a boolean output, 1/0. It outputs 1 if $H(m) = F_P(\sigma)$, otherwise, 0.

Three-Move Type Signature Scheme. Three-move type signature schemes are derived from three-move honest verifier zero-knowledge proofs. Fiat-Shamir signature scheme [8] and Schnorr signature scheme [18] belong to this type. The signing algorithm $\mathcal{S}^{\mathsf{T}}$, which takes a message m and a private key $\hat{s}$, produces a signature $\sigma = (s, c)$. It consists of three polynomial-time functions: A, H and Z. Function A generates the first-move commitment as $z = A(\hat{s}, r)$ where r is some random number. $H : \{0,1\}^* \to \Delta$ is a hash function which generates a non-interactive challenge as $c = H(m, z)$. The size of Δ is assumed to be in the same magnitude as that of the space of the public key pair $(\hat{s}, P)$. Z generates an answer $s = Z(\hat{s}, r, c)$. The verification algorithm $\mathcal{V}^{\mathsf{T}}$, which takes a message m, a signature $\sigma = (s, c)$ and a public key P, produces a boolean output. It involves a function V such that if the signature σ is valid and correct, then

$$c = H(m, V(P, s, c)).$$

6.1 The Generic Scheme

Let $L = \{P_1, \cdots, P_n\}$ be a set of n public keys corresponding to n possible signers. For $i = 1, \cdots, n$, let $a + b$ denote the group operation of Abelian group Δ_i and $a - b$ be the group operation with inverse of b, for all $a, b \in \Delta_i$. Domain Δ_i depends on F_{P_i} (trapdoor-one-way type) or P_i (three-move type).

Let ρ be twice of the largest $|\Delta_i|$, $1 \le i \le n$. Let $G : \{0,1\}^* \to \{0,1\}^\rho$ be some hash function. Without loss of generality, assume user i, for $t+1 \le i \le n$, are non-participating signers and user j, for $1 \le j \le t$, are participating signers. To generate a (t, n)-threshold ring signature on a message m, the participating signers with their secret keys $\{\hat{s}_1, \cdots, \hat{s}_t\}$ carry out the following steps.

Signing Algorithm.

1. For $i = t+1, \cdots, n$, pick $c_i \in_R \{0,1\}^\rho$ and $s_i \in_R \Delta_i$. Compute

$$z_i = \begin{cases} V(P_i, s_i, \theta_i(c_i)) & \text{if user } i \text{ has a Three-move Type key pair} \\ \theta_i(c_i) + F_{P_i}(s_i) & \text{if user } i \text{ has a Trapdoor One-Way Type key pair} \end{cases}$$

   where $\theta_i : \{0,1\}^* \to \Delta_i$ is some hash function which maps to some appropriate range.

2. For $j = 1, \cdots, t$, pick $r_j \in_R \Delta_j$ and compute
$$z_j = \begin{cases} A(\hat{s_j}, r_j) & \text{if user } j \text{ has a Three-Move Type public key pair} \\ r_j & \text{if user } j \text{ has a Trapdoor One-Way Type key pair} \end{cases}$$
3. Compute $c_0 = G(L, t, m, z_1, \cdots, z_n)$ and construct a polynomial f over $GF(2^\rho)$ such that $\deg(f) = n-t$, $f(0) = c_0$ and $f(i) = c_i$ for $i = t+1, \cdots, n$.
4. For $j = 1, \cdots, t$, compute $c_j = f(j)$ and
$$s_j = \begin{cases} Z(\hat{s}_j, r_j, \theta_j(c_j)) & \text{if user } j \text{ has a Three-Move Type key pair} \\ F^{-1}_{\hat{s}_j}(r_j - \theta_j(c_j)) & \text{if user } j \text{ has a Trapdoor One-Way Type key pair} \end{cases}$$
where $\theta_j : \{0,1\}^* \to \Delta_j$ is some hash function which maps to some appropriate range.
5. Output the signature for m and L as $\sigma = (s_1, \cdots, s_n, f)$.

Verification Algorithm. A verifier checks a signature $\sigma = (s_1, \cdots, s_n, f)$ with a message m and a set of public keys $L = \{P_1, \cdots, P_n\}$ as follows.

1. Check if $\deg(f) = n-t$. If yes, proceed. Otherwise, reject.
2. For $i = 1, \cdots n$, compute $c_i = f(i)$ and
$$z'_i = \begin{cases} V(P_i, s_i, \theta_i(c_i)) & \text{if user } i \text{ has a Three-move Type key pair} \\ \theta_i(c_i) + F_{P_i}(s_i) & \text{if user } i \text{ has a Trapdoor One-way Type key pair} \end{cases}$$
3. Check whether $f(0) \overset{?}{=} G(L, t, m, z'_1, \cdots, z'_n)$. If not, reject. Otherwise, accept.

7 Conclusion

In this paper, we describe a (t, n)-threshold ring signature scheme which is perfectly separable. A signature produced by this scheme can include an arbitrary mixture of public keys which are for various flavors of traditional signature schemes. A concrete scheme of supporting the mixture of RSA-based and DL-based public keys with various security parameters is also given. The scheme is shown to be signer ambiguous and existentially unforgeable against chosen message attacks under the random oracle model.

References

[1] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *Proc. ASIACRYPT 2002*, pages 415–432. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2501. 14, 15, 20

[2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM Press, 1993. 14, 19

[3] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Proc. CRYPTO 2002*, pages 465–480. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2442. 13, 14, 15

[4] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. full version. http://www.di.ens.fr/ +bresson, 2002. 14, 15 \bibitem{CamenischDa00}J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *Proc. ASIACRYPT 2000*, pages 331–345. Springer-Verlag, 2000. Lecture Notes in Computer Science No. 1976.

[5] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes. In *Proc. CRYPTO 99*, pages 413–430. Springer-Verlag, 1999. Lecture Notes in Computer Science No. 1666. 13

[6] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. CRYPTO 94*, pages 174–187. Springer-Verlag, 1994. Lecture Notes in Computer Science No. 839. 14, 15, 17

[7] Y. Desmedt. Some recent research aspects of threshold cryptography. In *Proc. First International Workshop on Information Security, ISW 97*, pages 158–173. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1196. 13

[8] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. CRYPTO 86*, pages 186–199. Springer-Verlag, 1987. Lecture Notes in Computer Science No. 263. 14, 21

[9] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International Journal on Information Security*, 1(1):36–63, 2001. 12

[10] J. Kilian and E. Petrank. Identity escrow. In *Proc. CRYPTO 98*, pages 169–185. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1462. 13, 19

[11] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable and culpable ring signatures. Manuscript, 2003. 25

[12] National Bureau of Standards FIPS Publication 186. *Digital Signature Standard*, 1994. 12

[13] K. Ohta and T. Okamoto. On concrete security treatment of signatures derived from identification. In *Proc. CRYPTO 98*, pages 354–369. Springer-Verlag, 1998. Lecture Notes in Computer Science No. 1462. 25

[14] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Proc. EUROCRYPT 96*, pages 387–398. Springer-Verlag, 1996. Lecture Notes in Computer Science No. 1070. 25

[15] M. O. Rabin. Digitalized signatures as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, January 1979. 21

[16] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *Proc. ASIACRYPT 2001*, pages 552–565. Springer-Verlag, 2001. Lecture Notes in Computer Science No. 2248. 13, 14, 15

[17] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. 12, 21

[18] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Proc. CRYPTO 89*, pages 239–252. Springer, 1990. Lecture Notes in Computer Science No. 435. 12, 14, 21

[19] H. C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, IT-26(6):726–729, November 1980. 21

[20] D. S. Wong, K. Fung, J. K. Liu, and V. K. Wei. On the RS-Code construction of ring signature schemes and a threshold setting of RST. In *5th Intl. Conference on Information and Communication Security (ICICS 2003)*, pages 34–46. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2836. 14, 15

[21] F. Zhang and K. Kim. ID-Based blind signature and ring signature from pairings. In *Proc. ASIACRYPT 2002*, pages 533–547. Springer-Verlag, 2002. Lecture Notes in Computer Science No. 2501. 14

A Proof of Lemma 1

Proof. Suppose on inputs of any set of $t-1$ private keys $\hat{S}_{t-1}$, any set of n public keys L, in which the corresponding public keys of the private keys in $\hat{S}_{t-1}$ are included, and any binary string $m \in \{0,1\}^*$, there exists a PPT E that outputs a forged signature $\sigma = (s_1, \cdots, s_n, f)$ such that $\deg(f) = n-t$ with non-negligible probability p_E. We construct from E a PPT B that solves DLP with non-negligible probability. That is, given (p, q, g, y), B outputs an integer x such that $g^x \equiv y \pmod p$ with non-negligible probability.

To get the black-box E run properly, B needs to provide a set of $t-1$ private keys $\hat{S}_{t-1}$, a set of n public keys L, in which the corresponding public keys of the private keys in $\hat{S}_{t-1}$ are included, and some binary string m as inputs. To do this, B randomly generates a set of $n-1$ public key pairs $\Pi = \{(x_i, (p_i, q_i, g_i, y_i))\}_{1 \le i \le n-1}$ where $x_i \in \mathbb{Z}_{q_i}$ and $y_i \equiv g_i^{x_i} \pmod{p_i}$. Next B randomly picks $t-1$ private keys from Π and denotes the set to be $\hat{S}_{t-1}$. Then B sets L to be the collection of all the public keys in Π and (p, q, g, y) and m to be an arbitrary binary string. Without loss of generality, let π index (p, q, g, y) in L.

Besides, B also simulates E's point of view by constructing the random oracle G and the signing oracle $\mathcal{SO}$. We first describe the construction of the signing oracle $\mathcal{SO}$. On inputs a set of public keys L where $n = |L|$, $i_1, \cdots, i_t$ such that $1 \le i_j \le n$ for $1 \le j \le t$ and $t \le n$, and some binary string $m \in \{0,1\}^*$, the answer is simulated as follows.

1. Randomly generate $c_0, c_{t+1}, \cdots, c_n \in_R \{0,1\}^\rho$.
2. Construct f over $GF(2^\rho)$ such that $\deg(f) = n-t$ and $f(0) = c_0, f(i) = c_i$, for $i = t+1, \cdots, n$.
3. Compute $c_1 = f(1), \cdots, c_t = f(t)$.
4. Randomly generate $s_i \in_R \mathbb{Z}_{q_i}$ for $i = 1, \cdots, n$.
5. Compute $z_i = g_i^{s_i} y_i^{c_i} \bmod p_i$ for $i = 1, \cdots, n$.
6. Assign c_0 as the value of $G(L, t, m, z_1, \cdots, z_n)$.
7. Output $(s_1, \cdots, s_n, f)$.

The simulation fails if step 6 causes collision, that is, the value of c_0 has been assigned before. This happens with probability at most $q_G/2^\rho$ where q_G is the number of times that the random oracle G is queried by E. Since E only queries G for polynomially number of times, the simulation is successful with overwhelming probability.

Let Θ, Ω be the random tapes given to the signing oracle and E such that E outputs a forged signature. Notice that the success probability of E is taken over the space defined by Θ, Ω and the random oracle G. The forged signature contains a polynomial f where $f(0) = G(L, t, m, z_1, \cdots, z_n)$ for $z_i \equiv g^{s_i} y_i^{f(i)} \pmod{p_i}$, $1 \leq i \leq n$. With probability at least $1 - 2^{-\rho}$, there exists a query $G(L, t, m, z_1, \cdots, z_n)$ due to the assumption of ideal randomness of G. Split G as (G^-, c_0) where G^- corresponds to the answers to all G-queries except for c_0. By invoking E with (Θ, Ω, G^-) and randomly chosen c_0' $(\neq c_0)$ polynomial times $\rho'(q_G)/p_E$, E outputs at least one forged signature $\sigma' = (s_1', \cdots, s_n', f')$ with probability $p_E/\rho''(q_G)$, due to the heavy-row lemma [13] where ρ' and ρ'' are some polynomial functions. Since (Θ, Ω, G^-) is fixed, z_1 to z_n are unchanged for both runs. Therefore, B can compute the discrete-log $x = (s_\pi - s_\pi')/(c_\pi' - c_\pi) \bmod q$.

To see that $c_\pi' \neq c_\pi$ (that is, $f'(\pi) \neq f(\pi)$), we notice that since $f'(0) \neq f(0)$ and the degrees of f and f' are limited to $n - t$, there is at least one value $1 \leq j \leq n$ such that $f'(j) \neq f(j)$. The chance of having $j = \pi$ is $1/n$ if π is randomly chosen. Hence the probability of having B succeed is at least $p_E/(n\rho''(q_G))$ which is non-negligible.

Notice that in the tape rewind simulation above, forking lemma [14] or ROS (Rewind-on-Success) lemma [11] can also be used to derive the success probability of machine B. □

B Proof of Lemma 2

Proof. Let $\mathcal{A}$ be a PPT adversary who can forge signatures with non-negligible probability at least $1/Q_1(k)$ when given a security parameter k, n RSA encryption functions and strictly less than t of the matching decryption functions where Q_1 is some polynomial. Assume $\mathcal{A}$ makes q_G queries to G, q_S queries to the signing oracle $\mathcal{SO}$, and a total of q_H queries to H_1, $\cdots$, H_n combined. We construct from $\mathcal{A}$ a PPT $\mathcal{M}$ that solves t copies of RSA-Collision with respect to t different encryption functions with non-negligible probability.

A successful forgery $\sigma = (s_1, \cdots, s_n, f)$ generated by $\mathcal{A}$ with Turing transcript $\mathcal{T}$ (including coin flip sequences) is called an ℓ-forgery if the first appearance in $\mathcal{T}$ of the query $G(L, t, m, z_1, \cdots, z_n)$ is the ℓ-th G-query, $1 \leq \ell \leq q_G + q_S$. Note $c_i = f(i)$, $0 \leq i \leq n$; $z_i = H_i(c_i) + s_i^{e_i}$, $1 \leq i \leq n$.

$\mathcal{M}$ simulates $\mathcal{A}$'s point of view by making G, H_1, $\cdots$, H_n independent random oracles, and implements the signing oracle $\mathcal{SO}$ by our signing algorithm similar to the construction described in the proof of Lemma 1. Additional specifications of $\mathcal{M}$ will be presented after some technical definitions.

For each ℓ, $1 \leq \ell \leq q_G + q_S$, $\mathcal{M}$ makes a simulation run described below. Random oracles are started anew for each pass.

In the ℓ-th pass, $\mathcal{M}$ simulates $\mathcal{A}$ to obtain $\sigma = (s_1, \cdots, s_n, f)$. Then it rewinds to the ℓ-th G-query and resimulates $\mathcal{A}$ from there onward to obtain $\bar{\sigma} = (\bar{s}_1, \cdots, \bar{s}_n, \bar{f})$.

There exists ℓ such that the probability of $\mathcal{A}$ generating an ℓ-forgery is at least $1/(Q_1(k)(q_G + q_S))$. By the heavy-row lemma, the probability of returning two successful ℓ-forgeries in the ℓ-th pass is at least $1/(4Q_1^2(k)(q_G + q_S)^2)$.

Consider the case where two successful ℓ-forgeries, denoted $\sigma = (s_1, \cdots, s_n, f)$ and $\bar{\sigma} = (\bar{s}_1, \cdots, \bar{s}_n, \bar{f})$, are returned in the ℓ-th pass of the simulation. The collision-freeness of H implies that $\bar{f}(0) \neq f(0)$, and $\bar{f} \neq f$. Since both polynomials have degree $n - t$, there exist at least t integers, $1 \leq i_1 < \cdots < i_t \leq n$, such that $f(i_j) \neq \bar{f}(i_j)$, $1 \leq j \leq t$. In the rewind simulation, the inputs to the ℓ-th G-query are identical to the first simulation. Therefore, $z_i = H_i(c_i) + s_i^{e_i} = H_i(\bar{c}_i) + \bar{s}_i^{e_i}$, $1 \leq i \leq n$. Therefore, $\mathcal{M}$ has obtained the T desired sets of RSA-Collisions $z_i = H_i(c_i) + s_i^{e_i} = H_i(\bar{c}_i) + \bar{s}_i^{e_i}$ and $(c_i, s_i) \neq (\bar{c}_i, \bar{s}_i)$, for $i \in \{i_1, \cdots, i_t\}$. The complexity of $\mathcal{M}$ is $q_G + q_S$ times that of $\mathcal{A}$, and the probability of success of $\mathcal{M}$ is at least $1/(4Q_1^2(k)(q_G + q_S)^2)$. □

Remark: We have proven that if $\mathcal{A}$ can solve RSA-Collision with non-negligible probability, then $\mathcal{A}$ can forge signatures. The technique will be discussed in another paper.

On the Security of a Group Signature Scheme with Forward Security

Guilin Wang

Infocomm Security Department (ICSD)
Institute for Infocomm Research (I^2R)
21 Heng Mui Keng Terrace, Singapore 119613
`http://www.i2r.a-star.edu.sg/icsd/staff/guilin`
`glwang@i2r.a-star.edu.sg`

Abstract. A group signature scheme allows a group member of a given group to sign messages on behalf of the group in an anonymous and unlinkable way. In case of a dispute, however, a designated group manager can reveal the signer of a valid group signature. Based on Song's forward-secure group signature schemes, Zhang, Wu, and Wang proposed a new group signature scheme with forward security at ICICS 2003. Their scheme is very efficient on both communication and computation aspects. Unfortunately, their scheme is *insecure*. In this paper we present a security analysis to show that their scheme is *linkable*, *untraceable*, and *forgeable*.

Keywords: digital signature, group signature, forward security, cryptanalysis.

1 Introduction

In modern electronic society, digital signatures are playing an important role to provide integrity, authentication and undeniability for electronic transactions. *Group signatures*, first introduced by Chaum and van Heyst in [14], are a special type of digital signatures. In such a scheme each group member of a given group is allowed to sign messages on behalf of the group in an anonymous and unlinkable way. To check the validity of a group signature, a receiver only needs to get the unique group public key. However, with the exception of a designated *group manager*, anybody else can neither identify the identity of the signer nor determine whether multiple signatures are generated by the same group member. Furthermore, in case of later disputes, the group manager can "open" a group signature and then reveal the true signer's identity.

From the viewpoints of verifiers, only a single group public key is needed to verify group signatures. On the other hand, from the viewpoint of the signing group, its internal structure is hidden from verifiers while the signers' identities can be revealed, if necessary. In virtue of these advantages, group signatures have many potential applications, such as e-voting, e-bidding and e-cash etc [15, 20, 23, 21, 13].

Following the first schemes constructed in [14], a number of new group signature schemes and improvements have been proposed [15, 9, 22, 10, 2, 3, 19, 8, 24, 12, 7]. In [15], Chen and Pedersen constructed the first scheme which allows new members to join the group dynamically, and suggested to use group signatures in e-bidding. Camenisch and Stadler proposed the first group signature scheme that can be used for large groups, since in their scheme the group public key and signatures have lengths independent of the group size [9]. Based on the strong RSA assumption [16], Camenisch and Michels presented an efficient group signature scheme in [10, 11]. Later, Kim et al. extended their scheme to support efficient member revocation [19]. Ateniese and Tsudik pointed out some obstacles that stand in the way of real world applications of group signatures, such as coalition attacks and member deletion [2]. In fact, there have been several papers which focused on the problem of member deletion [19, 8, 4]. Ateniese et al. presented a provably secure group signature scheme in [3].

Song addressed two important problems in group signature schemes, i.e., how to deal with exposure of group signing keys and how to efficiently revoke group members [24]. Here, a *group signing key* is referred to all secrets that enable a signer to produce group signatures in [24]. In fact, a group signing key consists of the membership secret and the group membership certificate [9, 3]. Based on the idea of forward secure signatures [1, 6, 17], Song constructed the fist two *forward-secure group signature* schemes. In such a scheme, the expected system life-time is divided into T time periods, and each group member's signing key evolves over time. In time period $j+1$, the signing key sk_{j+1} is updated from the signing key sk_j for time period j by using a public one-way function, and then sk_j is erased from the system. When the signing key sk_j is compromised, an attacker cannot sign messages with respect to any previous period. Especially, he cannot derive any previous signing key which corresponds to a time period i, $i < j$. Furthermore, Song also extended her schemes to support group member revocation. In Song's schemes, the group public key is affected neither by signing key update, nor by group member join or leave.

In [28], Zhang, Wu, and Wang proposed a newly efficient forward-secure group signature scheme. Since *signatures of knowledge* (refer to [9, 3]) are not used, their scheme is really very efficient on both computation and communication aspects. For example, the total computation cost of their signature generation and verification is only 7 modular exponentiations, while 36 modular exponentiations are needed in Song's schemes. At the same time, they claimed that their scheme satisfies all the desired security requirements (see Section 2). However, we find this is not the fact.

In this paper, we present a security analysis of Zhang-Wu-Wang group signature scheme with forward security [28]. More specifically, we demonstrate that their scheme is *linkable*, *untraceable* and *forgeable*. We first identify an effective way that allows anybody to determine whether two group signatures are signed by the same group member. Then, we demonstrate that any group member can forge valid signatures which cannot be opened by the group manager. This implies that their `OPEN` procedure fails to trace malicious group members. Furthermore, we prove that Zhang-Wu-Wang scheme is *untraceable in essence*, i.e., their scheme cannot meet the traceability if one just modifies `OPEN` procedure. Finally, under reasonable assumptions, a *universally forging attack* is presented. In this attack, even an outsider (not a group member) can forge valid group signatures on any messages of his choice. Therefore, Zhang-Wu-Wang scheme is *insecure*, though it is very efficient.

The rest of this paper is organized as follows. In Section 2, we introduce the informal definitions of a forward-secure group signature scheme and the security requirements. Section 3 reviews Zhang-Wu-Wang scheme [28]. Then, our security analysis is presented in Section 4. Finally, Section 5 concludes this paper.

2 Definitions

A forward-secure group signature scheme involves a *group manager*, a set of *group members*, and a set of *verifiers*. The group manager (for short, GM) is responsible for admitting/revoking group members, and for opening group signatures to reveal the true signers. When a potential user registers with GM, he/she becomes a group member and then can sign messages on behalf of the group. A verifier checks the validity of a group signature by using the unique group public key. The computational capability of each entity is modeled by a probabilistic polynomial-time Turing machine. We now review the definitions of forward-secure group signature schemes and their security requirements as follows. For more formal definitions on this subject, please refer to [7].

Definition 1. A forward-secure group signature scheme *is comprised of the following procedures [9, 2, 3, 24, 28]:*

- `SETUP`: *On input of a security parameter ℓ, this probabilistic algorithm outputs the initial group public key and the secret key for the group manager.*
- `JOIN`: *An interactive protocol between the group manager and a user that results in the user becoming a new group member. The user's output is a group signing key, i.e., a membership secret and the corresponding membership certificate.*
- `EVOLVE`: *An algorithm that on input a group signing key for time period j, outputs the corresponding group signing key for time period $j+1$.*
- `SIGN`: *A probabilistic algorithm that on input a group public key, a group signing key, and a message m outputs a group signature on m.*
- `VERIFY`: *An algorithm for establishing the validity of an alleged group signature of a message with respect to a group public key.*
- `OPEN`: *An algorithm that, given a message, a valid group signature on it, a group public key and the corresponding group manger's secret key, determines the identity of the signer.*
- `REVOKE`: *An algorithm that on input a group member's certificate, a group public key and the corresponding group manger's secret key, outputs a revocation token that revokes the group member's signing ability.*

Definition 2. *A forward-secure group signature scheme is* secure *if it satisfies all the following security requirements [2, 3, 24, 28]:*

- **Correctness***: Signatures produced by a group member using* `SIGN` *procedure must be accepted by* `VERIFY` *procedure.*
- *Unforgeability: Only group members are able to sign messages on behalf of the group.*
- **Anonimity***: Given a valid group signature for some message, identifying the actual signer is computationally hard for everyone but the group manager.*
- **Unlinkability***: Deciding whether two different valid signatures were generated by the same group member is computationally hard for everyone but the group manager.*
- **Excupability***: Even if the group manager and some of the group members collude, they cannot sign on behalf of non-involved group members.*
- **Traceability***: The group manager can always open a valid group signature using* `OPEN` *procedure and then identify the actual signer.*
- *Coalition-resistance: A colluding subset of group members cannot generate a valid group signature that cannot be traced by the group manager.*
- **Revocability***: The group manager can revoke a group member's signing ability so that this group member cannot produce a valid group signature any more after being deleted.*
- **Forward security***: When a group signing key is exposed, previously generated group signatures remain valid and do not need to be re-signed.*

3 Review of Zhang-Wu-Wang Scheme

3.1 `SETUP` Procedure

The group manager (GM, for short) randomly chooses two large primes p_1, p_2 of the same size such that $p_1 = 2p_1' + 1$ and $p_2 = 2p_2' + 1$, where both p_1' and p_2' are also primes. Let $n = p_1 p_2$, and $G = \langle g \rangle$ be a cyclic subgroup of $\mathbb{Z}_n^*$. Usually, G is selected as the set of quadratic residues modulo n [10, 3, 24], i.e., g's order $\mathrm{ord}(g) = p_1' p_2'$. GM randomly chooses an integer x as his secret key, and then sets his public key as $y = g^x \bmod n$. GM selects a random integer e (e.g., $e = 3$) which satisfies $\gcd(e, \phi(n)) = 1$, and computes d such that $de = 1 \bmod \phi(n)$. Let $h(\cdot)$ be a publicly known coalition-resistant hash function (e.g., SHA-1, MD5). The expected system life-time is divided into T intervals and the intervals are publicly known. $(c, s) = SPK\{\gamma : y = g^\gamma\}(\ )$ denotes the signature of knowledge of $\log_g y$ on the empty message (see [9, 3] for details). Finally, the group manager publishes the public key $(y, n, g, e, h(\cdot), ID_{GM}, T)$, where ID_{GM} is the identity of the group manager.

3.2 `JOIN` Procedure

If a user, say Bob, wants to join the group, he executes an interactive protocol with GM. Firstly, Bob chooses a random number $k \in \mathbb{Z}_n^*$ as his secret key, and computes his identity $ID_B = g^k \bmod n$. Then, Bob generates the signature of knowledge $(c, s) = SPK\{k : ID_B = g^k\}(\ )$ to show that he knows a secret value k to meet $ID_B = g^k \bmod n$. Finally, Bob keeps k privately and sends $(ID_B, (c, s))$ to the group manager.

Upon receiving $(ID_B, (c, s))$, GM first verifies the signature of knowledge (c, s). If the verification holds, GM chooses a random number $\alpha \in \mathbb{Z}_n^*$, and computes a triple (r_B, s_B, W_{B_0}) from

$$r_B = g^{\alpha} \bmod n, \ \ s_B = \alpha + r_B x, \ \ w_{B_0} = (r_B ID_{GM} ID_B)^{-d^T} \bmod n.$$

Then, GM sends Bob (s_B, r_B, w_{B_0}) via a private channel, and stores (s_B, r_B, w_{B_0}) together with $(ID_B, (c, s))$ in his local database. Bob accepts (s_B, r_B, w_{B_0}) as his initial membership certificate if the following two equalities hold.

$$g^{s_B} \equiv r_B y^{r_B} \bmod n, \quad \text{and} \quad r_B ID_{GM} ID_B \equiv {w_{B_0}}^{-e^T} \bmod n. \tag{1}$$

3.3 EVOLVE Procedure

Assume that Bob has the group membership certificate (s_B, r_B, w_{B_j}) at time period j. Then at time period $j+1$, he updates his group membership certificate as $(s_B, r_B, w_{B_{j+1}})$ by computing

$$w_{B_{j+1}} = (w_{B_j})^e \bmod n \ (= (r_B ID_{GM} ID_B)^{-d^{T-j}} \bmod n). \tag{2}$$

3.4 SIGN Procedure

Let (s_B, r_B, w_{B_j}) be Bob's group membership certificate at time period j. To sign a message m, Bob randomly chooses two numbers $q_1, q_2 \in \mathbb{Z}_n^*$, and computes z_1, u, r_1, r_2, r_3 as follows:

$$\begin{aligned} z_1 &= g^{q_1} y^{q_2} \bmod n, \\ u &= h(z_1, m), \\ r_2 &= w_{B_j}^{u} \bmod n, \\ r_1 &= q_1 + (s_B + k) \cdot u \cdot h(r_2) \text{ (in } \mathbb{Z}), \\ r_3 &= q_2 - r_B \cdot u \cdot h(r_2) \text{ (in } \mathbb{Z}). \end{aligned} \tag{3}$$

The resulting group signature on m is $\sigma = (u, r_1, r_2, r_3, m, j)$.

3.5 VERIFY Procedure

Given $\sigma = (u, r_1, r_2, r_3, m, j)$, a verifier accepts it as a valid group signature on m if and only if $u \equiv h(z_1', m)$, where z_1' is computed by

$$z_1' = ID_{GM}^{u \cdot h(r_2)} g^{r_1} r_2^{h(r_2) \cdot e^{T-j}} y^{r_3} \bmod n. \tag{4}$$

3.6 OPEN Procedure

Given a group signature $\sigma = (u, r_1, r_2, r_3, m, j)$, if necessary, GM can open it to reveal the actual identity of the signer who produced the signature. GM first checks σ's validity via VERIFY procedure. If σ is a valid signature, GM operates as follows to find the signer's identity:

(1) Compute $\eta = 1/(u \cdot h(r_2)) \bmod \phi(n)$.
(2) Compute $z_1' = ID_{GM}^{u \cdot h(r_2)} g^{r_1} r_2^{h(r_2) \cdot e^{T-j}} y^{r_3} \bmod n$.
(3) Search his database to find a pair (ID_B, r_B) that satisfies the following equality:
$$r_B ID_B \equiv (g^{r_1} y^{r_3} / z_1')^{\eta} \bmod n. \tag{5}$$
(4) If there is a tuple (r_B, ID_B) satisfying the above equation, GM concludes that ID_B is the identity of the actual signer. Otherwise, output $\perp$.

3.7 REVOKE Procedure

If GM wants to revoke Bob's membership certificate at time period j, he publishes a revocation token (R_j, j) in the CRL (the Certificate Revocation List), where the value R_j is computed by

$$R_j = (r_B ID_B)^{d^{T-j}} \bmod n. \tag{6}$$

Given a valid group signature $\sigma = (u, r_1, r_2, r_3, m, j)$, a verifier can identify whether σ is produced by a revoked group member. For this sake, he performs as follows:

(1) Compute $z_1' = ID_{GM}^{u \cdot h(r_2)} g^{r_1} r_2^{h(r_2) \cdot e^{T-j}} y^{r_3} \bmod n$.
(2) Search all (R_i, i) $(i \leqslant j)$ in CRL to check whether there is a R_i $(i \leqslant j)$ such that the following equality holds:
$$g^{r_1} y^{r_3} \equiv z_1' (R_i^{e^{T-i}})^{u \cdot h(r_2)} \bmod n. \tag{7}$$
(3) If one such R_i $(i \leqslant j)$ is found, the verifier concludes that the signature σ is revoked, i.e., it is generated by a group member after he is deleted.

4 Security Analysis of Zhang-Wu-Wang Scheme

In [28], Zhang et al. analyzed the security of their scheme, and concluded that their scheme satisfies all the security requirements listed in Section 2. However, we find that this is not the fact.

4.1 Linkability

Let $\sigma = (u, r_1, r_2, r_3, m, j)$ and $\bar{\sigma} = (\bar{u}, \bar{r}_1, \bar{r}_2, \bar{r}_3, \bar{m}, j)$ be two (valid) group signatures. To decide whether they are signed by the same group member, a verifier only needs to check whether the following equality holds.

$$r_2^{\bar{u}} \equiv \bar{r}_2^{u} \bmod n. \tag{8}$$

In fact, if both σ and $\bar{\sigma}$ are signed by the same group member, say Bob, according to `SIGN` procedure, we know that $r_2^{\bar{u}} = (w_{B_j})^{u \cdot \bar{u}} \bmod n = \bar{r}_2^{u} \bmod n$. So the above equality holds for σ and $\bar{\sigma}$.

On the other hand, we can show that if σ and $\bar{\sigma}$ are signed by two different group members, say Bob and Charlie, respectively, equation (8) unlikely holds. To prove this claim, on the contrary, we assume that σ and $\bar{\sigma}$ satisfy equation (8). Let $r_B = g^{\alpha} \bmod n$, $r_C = g^{\bar{\alpha}} \bmod n$, $ID_B = g^{k} \bmod p$, and $ID_C = g^{\bar{k}} \bmod p$. Since σ is signed by Bob, and $\bar{\sigma}$ is signed by Charlie, we have $r_2^{\bar{u}} = (w_{B_j})^{u \cdot \bar{u}} \bmod n$, and $\bar{r}_2^{u} = (w_{C_j})^{u \cdot \bar{u}} \bmod n$. From $r_2^{\bar{u}} = \bar{r}_2^{u} \bmod n$, we have $(r_B ID_{GM} ID_B)^{-u\bar{u}d^{T-j}} = (r_C ID_{GM} ID_C)^{-u\bar{u}d^{T-j}} \bmod n$. So, the following equation holds

$$g^{(k-\bar{k}+\alpha-\bar{\alpha})u\bar{u}d^{T-j}} = 1 \bmod n. \tag{9}$$

Since $\mathrm{ord}(g) = p_1' p_2'$, $\gcd(d, \phi(n)) = 1$, and $\phi(n) = 4p_1'p_2'$, we know $\gcd(d, \mathrm{ord}(g))$ $=1$. At the same time, usually $|h(\cdot)| = 160$, and $|p_1'| = |p_2'| \geqslant 255$, thus we also have $\gcd(u\bar{u}, \mathrm{ord}(g)) = 1$. Therefore, from equation (9), we conclude that $\mathrm{ord}(g) | (k - \bar{k} + \alpha - \bar{\alpha})$, i.e., $(k - \bar{k} + \alpha - \bar{\alpha}) = 0$ or $(k - \bar{k} + \alpha - \bar{\alpha}) = b \cdot \mathrm{ord}(g)$ for some non-zero integer b. However, both cases unlikely happen, because $\mathrm{ord}(g)$ is the product of two large primes, α and $\bar{\alpha}$ are random numbers selected by GM, and k and $\bar{k}$ are random numbers selected by Bob and Charlie, respectively.

Furthermore, equation (8) can be generalized to link two signatures which are generated by the same group member at different time periods. That is, given two group signatures $\sigma = (u, r_1, r_2, r_3, m, j)$ and $\bar{\sigma} = (\bar{u}, \bar{r}_1, \bar{r}_2, \bar{r}_3, \bar{m}, i)$ where $j > i$, one can know whether the same group member generated them by checking

$$r_2^{\bar{u}} \equiv \bar{r}_2^{u \cdot e^{j-i}} \bmod n. \tag{10}$$

4.2 Untraceability

In this section, we demonstrate an attack that enables a group member Bob to forge a valid certificate. Then, using this forged certificate, he can generate valid group signature on any message of his choice without being traced. Firstly, we note that it seems difficult to forge a new pair (r_B, s_B) so that the first equation in (1) is satisfied, since Bob does not know GM's secret key x. However, Bob can change the values of w_{B_0} and ID_B, and get a new certificate. For this sake, he chooses a random number $a \in \mathbb{Z}_n$, and defines $\bar{w}_{B_0}$, $\overline{ID}_B$ and $\bar{k}$ as follows:

$$\bar{w}_{B_0} = w_{B_0} g^{a} \bmod n,\ \overline{ID}_B = ID_B \cdot g^{-ae^{T}} \bmod n,\ \bar{k} = k - ae^{T} \text{ (in } \mathbb{Z}). \tag{11}$$

In the following, we show that the tuple $(s_B, r_B, \bar{w}_{B_0})$ with respect to $(\overline{ID}_B, \bar{k})$ constitutes a valid group membership certificate. Firstly, it is easy to know that $\overline{ID}_B = ID_B g^{-a\cdot e^T} \bmod n = g^{k-a\cdot e^T} \bmod n = g^{\bar{k}} \bmod n$. Secondly, we have $g^{s_B} = r_B y^{r_B} \bmod n$. Finally, the following equalities hold

$$\begin{aligned} r_B ID_{GM}\overline{ID}_B &= (r_B ID_{GM} ID_B)\cdot g^{-ae^T} \bmod n \\ &= {w_{B_0}}^{-e^T}\cdot g^{-ae^T} \bmod n \\ &= (w_{B_0}\cdot g^a)^{-e^T} \bmod n \\ &= \bar{w}_{B_0}^{-e^T} \bmod n. \end{aligned}$$

Therefore, according to JOIN procedure, Bob obtains another new certificate $(s_B, r_B, \bar{w}_{B_0})$ with $(\overline{ID}_B, \bar{k})$. Using this tuple $(s_B, r_B, \bar{w}_{B_0}, \overline{ID}_B, \bar{k})$, Bob can generate valid group signatures on arbitrary messages. According to OPEN procedure, when such forged signatures are presented, Bob will not be traced as the signer since $r_B\overline{ID}_B \neq r_B ID_B \bmod n$. Therefore, the OPEN procedure provided by [28] fails to trace such malicious group members. A natural question is whether we can improve this OPEN procedure such that it can reveal the identities of malicious group members. Unfortunately, the answer is negative. In other words, Zhang-Wu-Wang scheme is *untraceable in essence*, i.e., two malicious members may forge the same valid but untraceable group signature on a given message. More formally, we have the following theorem.

Theorem 1. *Using the above attack, the forged group signatures generated by two malicious members for the same message are perfectly indistinguishable.*

Proof: We only need to prove that if Bob forges a group signature σ on a message m, Charlie can also generate it by using our above attack. For simplicity, let $sk_{B,j} = (s_B, r_B, w_{B_j}, ID_B, k)$, and $fsk_{B,j} = (s_B, r_B, \bar{w}_{B_j}, \overline{ID}_B, \bar{k})$, where $w_{B_j} = w_{B_{j-1}}^e \bmod n = w_{B_0}^{e^j} \bmod n$, and $\bar{w}_{B_j} = \bar{w}_{B_{j-1}}^e \bmod n = \bar{w}_{B_0}^{e^j} \bmod n$. $sk_{B,j}$ and $fsk_{B,j}$ denote Bob's *signing key* and *forged signing key* at time period j, respectively. Here, $\bar{w}_{B_0}, \overline{ID}_B$ and $\bar{k}$ are computed by Bob as in the above attack, i.e., Bob selects a random number a and calculates the values of them from equation (11). To forge a group signature on message m, he randomly chooses two numbers $q_1, q_2 \in_R \mathbb{Z}_n^*$, and computes $z_1 = g^{q_1}y^{q_2} \bmod n$, $u = h(z_1, m)$, $r_2 = \bar{w}_{B_j}^u \bmod n$, $r_1 = q_1 + (s_B + \bar{k})uh(r_2)$ (in $\mathbb{Z}$), and $r_3 = q_2 - r_B uh(r_2)$ (in $\mathbb{Z}$). $\sigma = (u, r_1, r_2, r_3, m, j)$ is the resulting forged group signature on message m.

Now, we show that Charlie can forge a group signature σ' on the same message m such that $\sigma' \equiv \sigma$, if he chooses an appropriate number a' to define his forged signing key, and two specific numbers q_1' and q_2' to produce group signature. Let $sk_{C,j} = (s_C, r_C, w_{C_j}, ID_C, k')$ be Charlie's signing key at time period j, where $s_C = \alpha' + r_C x$, $r_C = g^{\alpha'} \bmod n$, $ID_C = g^{k'} \bmod n$, $w_{C_j} = w_{C_0}^{e^j} \bmod n$, and $w_{C_0} = (r_C ID_{GM} ID_C)^{-d^T} \bmod n$. To forge a new membership certificate, Charlie first sets $a' = a - (k + \alpha - k' - \alpha')\cdot e^{-T} \bmod \mathrm{ord}(g)$. This means that there exists an integer l such that $k' + \alpha' - a'e^T = k + \alpha - ae^T + l\cdot \mathrm{ord}(g)$. And then, according to equation (11), he defines $\bar{w}_{C_0}, \overline{ID}_C$ and $\bar{k}'$ as follows:

$$\bar{w}_{C_0} = w_{C_0} g^{a'} \bmod n,\ \overline{ID}_C = ID_C g^{-a'e^T} \bmod n,\ \bar{k}' = k' - a'e^T \text{ (in } \mathbb{Z}). \quad (12)$$

Charlie thus obtains his forged signing key $fsk_{C,j} = (s_C, r_C, \bar{w}_{C_j}, \overline{ID}_C, \bar{k}')$ for time period j, where $\bar{w}_{C_j} = \bar{w}^e_{C_{j-1}} \bmod n = \bar{w}^{e^j}_{C_0} \bmod n$. Due to the specific choice of the value of a', we have

$$r_B\overline{ID}_B = r_C\overline{ID}_C \bmod n, \quad \text{and} \quad \bar{w}_{B_0} = \bar{w}_{C_0} \bmod n.$$

To forge a group signature σ' on message m such that $\sigma' \equiv \sigma = (u, r_1, r_2, r_3, m, j)$, Charlie first sets $q_1' = q_1 + xuh(r_2)(r_B - r_C) - l \cdot uh(r_2) \cdot \mathrm{ord}(g)$, and $q_2' = q_2 - uh(r_2)(r_B - r_C)$. Then, he computes $z_1' = g^{q_1'}y^{q_2'} \bmod n$, $u' = h(z_1', m)$, $r_2' = \bar{w}^{u'}_{C_j} \bmod n$, $r_1' = q_1' + (s_C + \bar{k}')u'h(r_2')$ (in $\mathbb{Z}$), and $r_3' = q_2' - r_C u'h(r_2')$ (in $\mathbb{Z}$). Let $\sigma' = (u', r_1', r_2', r_3', m, j)$ be the resulting group signature forged by Bob. Then, one can directly verify that $z_1' = z_1$, $u' = u$, $r_1' = r_1$, $r_2' = r_2$ and $r_3' = r_3$. In other words, $\sigma' \equiv \sigma$.

The above discussion shows that for a given forged group signature σ, any group member may be the attacker who forged it by using our attack. Therefore, Theorem 1 holds[1]. $\square$

4.3 Forgeability

The attack given in Section 4.2 only enables group members to forge valid group signatures. This section demonstrates a universally forging attack which can be mounted by anybody, not necessarily a group member. We first describe our attack when the value of $ID_{GM}^{-d^{T-j}} \bmod n$ is available, and then explain how to get the value of $ID_{GM}^{-d^{T-j}} \bmod n$ using some public information.

If the value of $ID_{GM}^{-d^{T-j}} \bmod n$ is known, to forge a group signature on an arbitrary message m, the following *universally forging attack* can be mounted by anybody.

(1) Select four random numbers $a, b, q_1, q_2 \in_R \mathbb{Z}_n^*$.
(2) Compute $z_1 = g^{q_1}y^{q_2} \bmod n$, and $u = h(z_1, m)$.
(3) Define $r_2 = (ID_{GM}^{-d^{T-j}})^u g^{-a} y^b \bmod n$, $r_1 = q_1 + ah(r_2)e^{T-j}$ (in $\mathbb{Z}$), and $r_3 = q_2 - bh(r_2)e^{T-j}$ (in $\mathbb{Z}$).
(4) Output $\sigma = (u, r_1, r_2, r_3, m, j)$ as the forged group signature on message m.

We now show the correctness of our above attack. According to `VERIFY` procedure, we need to first compute z_1' and then check whether $u \equiv h(z_1', m)$. By equation (4), we have the following equalities:

$$\begin{aligned} z_1' &= ID_{GM}^{uh(r_2)} g^{r_1} r_2^{h(r_2)e^{T-j}} y^{r_3} \bmod n \\ &= ID_{GM}^{uh(r_2)} g^{r_1} ((ID_{GM}^{-d^{T-j}})^u g^{-a} y^b)^{h(r_2)e^{T-j}} y^{r_3} \bmod n \\ &= g^{r_1 - ah(r_2)e^{T-j}} y^{r_3 + bh(r_2)e^{T-j}} \bmod n \\ &= g^{q_1} y^{q_2} \bmod n \\ &= z_1 \bmod n. \end{aligned}$$

[1] Note that in the proof of Theorem 1, we only discuss the *possibility* whether two group members can forge the same valid but untraceable group signature on the same message, so Charlie can 'use' some secrets controlled by other parties, such as the values of a, q_1, q_2, x and $\mathrm{ord}(g)$ etc.

So, $z_1' \equiv z_1$. This means $u = h(z_1, m) = h(z_1', m)$, i.e., our above attack succeeds.

Now, we describe an algorithm which enables an outsider Alice to derive the value of $ID_{GM}^{-d^{T-j}} \bmod n$ alone from a number of known group signatures, revocation tokens and the group public key. Before presenting the details, we first explain the basic idea. Assume that the attacker Alice obtains two group signatures $\sigma = (u, r_1, r_2, r_3, m, i)$ and $\sigma' = (u', r_1', r_2', r_3', m', i)$, which are generated by Bob at time period i and satisfy $\gcd(u, u') = 1$. Later, GM revokes Bob's signing ability by releasing a revocation token (R_j, j) at time period j $(i < j)$. Since $\gcd(u, u') = 1$, by using extended Euclidian algorithm Alice can get two integers a and b such that $au + bu' = 1$. Then, she gets the value of w_{B_i} by

$$w_{B_i} = (r_2)^a \cdot (r_2')^b \bmod n. \tag{13}$$

Equation (13) holds because we have $w_{B_i} = w_{B_i}^1 \bmod n = w_{B_i}^{au+bu'} \bmod n = (w_{B_i}^u)^a (w_{B_i}^{u'})^b \bmod n = (r_2)^a \cdot (r_2')^b \bmod n$.

Finally, Alice computes the value of $ID_{GM}^{-d^{T-j}} \bmod n$ by

$$ID_{GM}^{-d^{T-j}} = (w_{B_i})^{e^{j-i}} \cdot R_j \bmod n. \tag{14}$$

Equation (14) is justified by the following equalities:

$$\begin{aligned} ID_{GM}^{-d^{T-j}} &= (ID_{GM} r_B ID_B)^{-d^{T-j}} \cdot (r_B ID_B)^{d^{T-j}} \bmod n \\ &= (ID_{GM} r_B ID_B)^{-d^{T-i} \cdot e^{j-i}} \cdot R_j \bmod n \\ &= (w_{B_i})^{e^{j-i}} \cdot R_j \bmod n. \end{aligned}$$

In the following complete description of our algorithm, Alice also has to find who is the signer revoked by (R_j, j).

Step (1). The attacker Alice collects a number of valid group signatures, and uses the method described in Section 4.1 to classify them into different directories, denoted by D_l $(1 \leqslant l \leqslant n)$. Here, the signatures in each directory are generated by a different signer, i.e., a different group member. For future use, she stores all classified signatures in her local database.

Step (2). Once the group manager releases a revocation token (R_j, j), she computes a value $P = R_j^{e^{T-j}} \bmod n$. Note that if Bob is the group member revoked by (R_j, j), it is easy to know that $P = r_B ID_B \bmod n$.

Step (3). Alice chooses one signature $\sigma = (u, r_1, r_2, r_3, m, i)$ from each non-empty directory D_l, and searches which signature satisfies the following equality:

$$r_2^{e^{T-i}} \equiv (ID_{GM} \cdot P)^{-u} \bmod n.$$

If one such signature $\sigma \in D_{\bar{l}}$ is found, Alice concludes that the group member (say Bob) corresponding to the directory $D_{\bar{l}}$ is deleted.

Step (4). Alice searches the directory $D_{\bar{l}}$ in her local database to find two group signatures $\sigma = (u, r_1, r_2, r_3, m, i)$, and $\sigma' = (u', r_1', r_2', r_3', m', i')$ such that $i' = i < j$ and $\gcd(u', u) = 1$. If two such signatures (generated by

Bob) are found, Alice executes the expended Euclidian algorithm to get two integers a and b such that $au + bu' = 1$. Then, Alice can get the value of w_{B_i} $(i < j)$ from equation (13).

Step (5). Finally, by using the values of w_{B_i} $(i < j)$ and R_j, Alice derives the value of $ID_{GM}^{-d^{T-j}} \bmod n$ by equation (14).

The success of the above algorithm depends on the following assumption: Alice can find two group signatures σ and σ' such that $\gcd(u', u) = 1$, and that they are generated during the same time period by the same group member who is deleted later. There are several reasons to support that this is a reasonable assumption in practical applications. Firstly, a group member of course may generate a number of signatures during the same time period, and be deleted later. Secondly, for two randomly selected integers N and M, $\gcd(N, M) = 1$ happens with probability $6/\pi^2 \approx 0.6$ [25, 29]. Since the hash function $h(\cdot)$ can be viewed as a random function with fixed bit-length output (e.g. 160-bit), u' and u can be treated as random numbers. Under this treatment, $\gcd(u', u) = 1$ also holds with probability about 0.6. Finally, when the value of $ID_{GM}^{-d^{T-j}} \bmod n$ is available, the attacker can use it to forge signatures for any time period i where $i \geq j$.

5 Conclusion

In this paper, we presented a security analysis of Zhang-Wu-Wang group signature scheme proposed in [28]. By successfully identifying several attacks, we demonstrated that their scheme is *insecure*. More specifically, our results show that their scheme is *linkable*, *untraceable* and *forgeable*. In fact, how to design a secure and more efficient group signature scheme is still a hot topic. The most recent investigations are given in [5, 7, 18, 26], and more literatures on group signatures are available at [27].

Acknowledgements

The author would like to thank Dr. Jianying Zhou, and the anonymous referees for their helpful comments and suggestions.

References

[1] R. Anderson. Invited Lecture, *4th ACM Computer and Communications Security*, 1997. 28

[2] G. Ateniese and G. Tsudik. Some open issues and new directions in group signature schemes. In: *Financial Cryptography (FC'99)*, LNCS 1648, pages 196-211. Springer-Verlag, 1999. 28, 29, 30

[3] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In: *Advances in Cryptology - CRYPTO 2000*, LNCS 1880, pages 255-270. Springer-Verlag, 2000. 28, 29, 30

[4] G. Ateniese, D. Song, and G. Tsudik. Quasi-efficient revocation of group signatures. In: *Financial Cryptography (FC'02)*, LNCS 2357. Springer-Verlag, 2002. Primary version available at `http://eprint.iacr.org/2001/101/` 28

[5] G. Ateniese and B. de Medeiros. Efficient group signatures without trapdoors. In: *Advances in Cryptology - ASIACRYPT 2003*, LNCS 2894, pages 246-268. Springer-Verlag, 2003. Primary version available at `http://eprint.iacr.org/2002/173/` 37

[6] M. Bellare, and S. Miner. A forward-secure digital signature scheme. In: *Advances in Cryptology - CRYPTO'99*, LNCS 1666, pages 431-448. Springer-Verlag, 1999. 28

[7] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: *Advances in Cryptology - EUROCRYPT'03*, LNCS 2656, pages 614-629. Springer-Verlag, 2003. 28, 29, 37

[8] E. Bresson and J. Stern. Efficient revocation in group signatures. In: *Public Key Cryptography (PKC'01)*, LNCS 1992, pages 190-206. Springer-Verlag, 2001. 28

[9] J. Camenisch and M. Stadler. Effient group signature schemes for large groups. In: *Advances in Cryptology - CRYPTO'97*, LNCS 1294, pages 410-424. Springer-Verlag, 1997. 28, 29, 30

[10] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In: *Advances in Cryptology - ASIACRYPT'98*, LNCS 1514, pages 160-174. Springer-Verlag, 1998. 28, 30, 38

[11] J. Camenisch and M. Michels. A group signature scheme based on an RSA-variant. *Technical Report RS-98-27*, BRICS, University of Aarhus, November 1998. An earlier version appears in [10]. 28

[12] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In: *Advances in Cryptology - CRYPTO 2002*, LNCS 2442, pages 61-76. Springer-Verlag, 2002. 28

[13] S. Canard and J. Traoré. On fair e-cash systems based on group signature schemes. In: *Information Security and Privacy (ACISP 2003)*, LNCS 2727, pages 237-248. Springer-Verlag, 2003. 27

[14] D. Chaum and E. van Heyst. Group signatures. In: *Advances in Cryptology - EUROCRYPT'91*, LNCS 950, pages 257-265. Springer-Verlag, 1992. 27, 28

[15] L. Chen and T. P. Pedersen. New group signature schemes. In: *Advances in Cryptology - EUROCRYT'94*, LNCS 950, pages 171-181. Springer-Verlag, 1995. 27, 28

[16] E. Fujisaki and T. Okamoto. Statistical zero-knowledge protocols to prove modular polynomial relations. In: *Advances in Cryptology - CRYPTO'97*, LNCS 1294, pages 16-30. Springer-Verlag, 1997. 28

[17] G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. In: *Advances in Cryptology - CRYPTO'01*, LNCS 2139, pages 332-354. Springer-Verlag, 2001. 28

[18] A. Kiayias and M. Yung. Extracting group signatures from traitor tracing schemes. In: *Advances in Cryptology - EUROCYPT 2003*, LNCS 2656, pp. 630-648. Springer-Verlag, 2003. 37

[19] H. J. Kim, J. I. Lim, and D. H. Lee. Efficient and secure member deletion in group signature schemes. In: *Information Security and Cryptology (ICISC 2000)*, LNCS 2015, pages 150-161. Springer-Verlag, 2001. 28

[20] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In: *Financial Cryptography (FC'98)*, LNCS 1465, pages 184-197. Springer-Verlag, 1998. 27

[21] G. Maitland and C. Boyd. Fair electronic cash based on a group signature scheme In: *Information Security and Cryptography (ICICS'01)*, LNCS 2229, pages 461-465. Springer-Verlag: 2001. 27

[22] H. Petersen. How to convert any digital signature scheme into a group signature scheme. In: *Security Protocols Workshop*, LNCS 1361, pages 177-190. Springer-Verlag, 1997. 28

[23] K. Sakurai and S. Miyazaki. An anonymous electronic bidding protocol based on a new convertible group signature scheme. In: *Information Security and Privacy (ACISP'00)*, LNCS 1841, pages 385-399. Springer-Verlag, 2000. 27

[24] D. X. Song. Practical forward secure group signature schemes. In: *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 225-234. ACM press, 2001. 28, 29, 30

[25] G. Tenenbaum. *Introduction to Analytic and Probabilistic Number Theory* (Theorem 5, page 41). Cambridge studies in advanced mathematics, Vol. 46. Cambridge University Press, 1995. 37

[26] G. Tsudik and S. Xu. Accumulating composites and improved group signing. In: *Advances in Cryptology -ASIACRYPT 2003*, LNCS 2894, pages 269-286. Springer-Verlag, 2003. Primary version available at `http://eprint.iacr.org/2003/112/` 37

[27] G. Wang. Bibliography on group signatures. `http://www.i2r.a-star.edu.sg/icsd/staff/guilin/bible/group-sign.htm` 37

[28] J. Zhang, Q. Wu, and Y. Wang. A novel efficient group signature scheme with forward security. In: *Information and Communications Security (ICICS'03)*, LNCS 2836, pages 292-300. Springer-Verlag, 2003. 28, 29, 30, 32, 34, 37

[29] Relatively Prime. `http://mathworld.wolfram.com/RelativelyPrime.html` 37

An Efficient Strong Designated Verifier Signature Scheme

Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch

Université Libre de Bruxelles
Département d'Informatique
Bd du Triomphe - CP212, 1050 Bruxelles, Belgium
{saeednia,skremer,omarkow}@ulb.ac.be

Abstract. This paper proposes a designated verifier signature scheme based on the Schnorr signature and the Zheng signcryption schemes. One of the advantages of the new scheme compared with all previously proposed schemes is that it achieves the "strong designated verifier" property without encrypting any part of the signatures. This is because the designated verifier's secret key is involved in the verification phase. Another advantage of the proposed scheme is the low communication and computational cost. Generating a signature requires only one modular exponentiation, while this amount is two for the verification. Also, a signature in our scheme is more than four times shorter than those of known designated verifier schemes.

Keywords. Signature, Designated verifier, Signcryption, Discrete logarithm.

1 Introduction

In 1989, Chaum and van Antwerpen [4] introduced the notion of undeniable signatures with the goal of enabling signers to have complete control over their signatures. That is, the verification of such signatures requires the participation of the signer (by means of an interactive protocol), in order to avoid undesirable verifiers getting convinced of the validity of the signatures. However, these signatures do not always achieve their goal, because of blackmailing [6, 7] and mafia attacks [5]. The problem is due to the fact that the signer does not know to whom s/he is proving the validity of a signature.

This weakness of undeniable signatures motivated the parallel introduction of designated verifier signatures by Jakobsson, Sako and Impagliazzo [8], as well as private signatures by Chaum [3]. Both of these signatures are based on the same idea and are nearly identical. In the rest of this paper, we only focus on the Jakobsson et al. scheme as it resulted in an academic publication. Designated verifier signatures provide authentication of a message, without however having the non-repudiation property of traditional signatures. In other words, they convince one—and only one—specified recipient that they are valid, but unlike standard digital signatures, nobody else can be convinced about their validity or invalidity. The reason is that the designated verifier in these schemes is able to create a signature intended to himself that is indistinguishable from a "real" signature. Therefore, when Bob receives a signature from Alice, he will certainly trust that it is originated from Alice upon verifying it, since he knows that he has not generated it himself. However, another party, Cindy, has no reason to accept such a signature as Alice's one, because she knows that Bob is fully capable to produce it himself.

Designated verifier signatures are very useful in various situations where the signer of a message should be able to specify who may be convinced by his/her signature. Let us consider the following example.

Suppose that a public institution initiates a call for tenders, asking some companies to propose their prices for a set of instruments and tasks to be accomplished. The institution may require the companies to sign their offers in order to make sure that they are actually authentic and originated from whom they claim to be. This is a valid requirement, but no company involved in this process desires its offer to affect other tenderers' decisions. That is, a company may capture a competitor's signed offer on the transmission line (to the institution) and prepares its offer consequently in order to increase its chance to be selected by the institution.

To prevent this, the companies may obviously encrypt their offers and signatures in order that they may only be read and verified by the institution. But, nothing prevents the latter to reveal them once decrypted. Indeed, since the institution's goal is to obtain a good price (as low as possible), it could show some signed offers to some other companies to influence them in making "good" offers.

So, the here raised question is about the conflict between authenticity and privacy. Designated verifier signatures are a solution to this problem. With such signatures, while the institution is convinced about the origin and the authenticity of an offer, it cannot transfer this conviction to others.

In 1996, Jakobsson et al. [8] proposed a designated verifier signature scheme as a non-interactive designated verifier proof of undeniable signatures (see section 3 for a description and the appendix for our cryptanalysis of this scheme). More recently, in 2001, Rivest, Shamir and Tauman introduced ring signatures [11], allowing to generate a signature linked to a group of potential signers. A special case of these signatures (by setting the size of the group to two) provides designated verifier signatures (see section 3).

Although these schemes are *signer ambiguous*, in the sense that one cannot verify whether the signer or the designated verifier issued the signature, they remain universally verifiable, i.e. anybody can make sure that there are only two potential signers. Hence, considering again the example above, if the companies' offers are sent just being signed using these designated verifier schemes, the signatures may be captured on the line before arriving at the institution, so that one can identify the signer, since it is now sure that the institution did not forge the signature. As indicated before, one possible solution, that is however expensive in terms of computational cost, is to encrypt each signature with the designated verifier's public key. This stronger requirement, called *strong designated verifier*, was briefly discussed in [8].

In this paper, we introduce a new efficient designated verifier signature scheme that is based on a combination of the Schnorr signature [12] and the Zheng signcryption schemes [13]. It requires only 1 modular exponentiation to generate and 2 modular exponentiations to verify a signature, i.e. no additional exponentiations are needed to convert the original Schnorr or Zheng schemes into a designated verifier signature scheme. Moreover, our scheme directly provides the strongness property without requiring any encryption of the signatures. This is particularly interesting in terms of computational cost as we will see further in this paper. Finally, the signatures in our scheme are very small in size.

The paper is organised as follows. In section 2, we recall definitions of designated verifier proofs of [8] and we give new definitions that we believe more suitable for our further analysis. In section 3, we briefly recall the existing designated verifier signature schemes. Section 4 presents the new scheme and section 5 discusses its security. In section 6, we consider the strong designated verifier property of our scheme and in section 7 we see a comparison with other designated verifier schemes in terms of performance. Finally, we show how to break and repair the Jakobsson et al. scheme in the appendix.

2 Definitions

Our goal is to allow Alice proving the validity of a statement Ω to Bob in such a way that, while Bob is convinced of this fact, he cannot transfer this conviction to other people.

As suggested in [8], when Alice wants to convince Bob—and only Bob—of the truth of the statement Ω, she should prove the statement "$\Omega \vee$ I know Bob's secret key". Bob, who is aware that he has not prepared the proof himself and knows that Alice does not know his secret key, will accept the validity of the first part of the statement (i.e., Ω) while no other verifier will be able to decide which part of the disjunction is true.

Informal definitions of the designated verifier proofs are given in [8]. We believe that these definitions, though completely persuasive, do not fully capture our intuition of the designated verifier proofs. Hereafter, we recall them and give more intuitive definitions that would be more suitable for further analysis of such schemes.

Definition 1 (Designated Verifier).
Let (P_A, P_B) be a protocol for Alice to prove the truth of the statement Ω to Bob. We say that Bob is a designated verifier if for any protocol (P_A, P'_B, P_C) involving Alice, Bob and Cindy, by which Bob proves the truth of ϑ to Cindy, there is another protocol (P''_B, P_C) such that Bob can perform the calculations of P''_B, and Cindy cannot distinguish transcripts of (P_A, P'_B, P_C) from those of (P''_B, P_C).

This definition clearly tells us that if Bob, after having received a proof (signature) from Alice, has a way to prove to Cindy the truth of a given statement, then he can produce indistinguishable transcripts by his own. As a consequence, whatever Bob can do with the "real" transcripts, he will be able to do with the "simulated" transcripts as well. Thus, Cindy being aware of this fact, will never be convinced by Bob's proof, whatever the protocol that Bob initiates.

Put in more formal words, we can define designated verifier proofs as follows:

Definition 2 (New Designated Verifier).
Let $P(A, B)$ be a protocol for Alice to prove the truth of the statement Ω to Bob. We say that Bob is a designated verifier if he can produce identically distributed transcripts that are indistinguishable from those of $P(A, B)$.

This is very close to the definition of zero-knowledge proofs by means of a simulator, except that a designated verifier proof is simulable with "no rewinding" of the simulator and especially with any challenge size. This precisely means that, in terms of signatures, the designated verifier signatures are the only kind of signature schemes that are zero-knowledge for the intended verifier. No classical signature scheme providing non-repudiation, including those derived from zero-knowledge identification schemes by means of a hash function, may be zero-knowledge, otherwise it is insecure.

Strong designated verifier proofs. In some circumstances, Cindy may be convinced with high probability that a designated verifier proof intended to Bob is actually generated by Alice, as Bob would not or could not generate it himself. For example:

1. When Bob is believed to be honest, Cindy would trust that Bob does never deviate from his prescribed protocol, so that by seeing a signature, she would be convinced that it is originated by Alice.
2. When Cindy is sure that Bob has not yet seen a signature intended to himself, she would be convinced that the signature is not "forged" by Bob.

In these cases, we need a stronger notion of designated verifier proofs that is defined in [8] as follows:

Definition 3 (Strong Designated Verifier).
Let (P_A, P_B) be a protocol for Alice to prove the truth of the statement Ω to Bob. We say that Bob is a strong designated verifier if for any protocol (P_A, P_B, P_D, P_C) involving Alice, Bob, Dave and Cindy, by which Dave proves the truth of ϑ to Cindy, there is another protocol (P'_D, P_C) such that Dave can perform the calculations of P'_D, and Cindy cannot distinguish transcripts of (P_A, P_B, P_D, P_C) from those of (P'_D, P_C).

Here again, all this definition amounts to saying is that the transcripts of a "real" proof may be simulated by anybody in such a way that they are indistinguishable for everybody other than Bob[1]. So, accordingly to our definition of designated verifier proofs, we define the strongness as follows:

Definition 4 (New Strong Designated Verifier).
Let $P(A,B)$ be a protocol for Alice to prove the truth of the statement Ω to Bob. We say that $P(A,B)$ is a strong designated verifier proof if anybody can produce identically distributed transcripts that are indistinguishable from those of $P(A,B)$ for everybody, except for Bob.

3 Related Work

In this section, we first recall the designated verifier signature scheme [8] introduced by Jakobsson, Sako and Impagliazzo[2] (JSI, for short). Then we present a special case of the ring signature scheme [11] introduced by Rivest, Shamir and Tauman (RST, for short) that provides the designated verifier property[3].

Another scheme that may be turned into a designated verifier signature is due to Abe, Ohkubo and Suzuki [1] and is known as 1-out-of-n signature. Because of lack of space, we do not describe this scheme. We just notice that it is essentially some kind of ring signatures that may make use of different type of keys.

3.1 The JSI Designated Verifier Scheme

The JSI scheme is a non-interactive designated verifier proof of Chaum's undeniable signatures. Let p be a large prime and q a prime divisor of $p-1$. Let g be a generator in $\mathbb{Z}_p^*$ of order q. A user u's private key will be denoted $x_u \in \mathbb{Z}_q^*$ and the corresponding public key $y_u = g^{x_u} \mod p$.

Signature generation. Alice wants to prove to Bob that $s = m^{x_a} \mod p$ is her signature on the message m. To prepare the poof, she selects w, r, t in $\mathbb{Z}_q$ and computes

$$\begin{aligned} c &= g^w y_b^r \mod p \\ G &= g^t \mod p \\ M &= m^t \mod p \\ h &= hash_q(c,G,M) \\ d &= t + x_a(h+w) \mod q \end{aligned}$$

where $hash_q$ denotes a hash function mapping values into $\mathbb{Z}_q$. Alice's signature on m and the associated proof is $\sigma = (s, w, r, G, M, d)$.

[1] Bob is actually the only party that can distinguish between real and simulated proofs. Even Alice cannot do so, if we assume that she does not keep the track of the proofs she generates.

[2] Chaum introduced a very similar scheme [3] under the name of private signatures.

[3] The main motivation of ring signatures is something more general than designing designated verifier signatures. The designated verifier property is rather a side effect of a particular setting of ring signatures.

Signature verification. Bob can verify that σ is a valid signature on the message m by computing

$$\begin{aligned} c &= g^w y_b^r \mod p \\ h &= hash_q(c, G, M) \end{aligned}$$

and by checking whether

$$\begin{aligned} G y_a^{h+w} &= g^d \mod p \\ M s^{h+w} &= m^d \mod p. \end{aligned}$$

If this holds, then Bob is convinced that Alice has generated this signature. However, other people (who can verify the consistency of the signature as Bob does) cannot conclude that Alice issued the signature, because they know that Bob can produce identically distributed transcripts, as follows.

Transcript simulation. To simulate transcripts of an Alice's signature, Bob chooses $d, \alpha, \beta \in \mathbb{Z}_q$ and computes

$$\begin{aligned} c &= g^\alpha \mod p \\ G &= g^d y_a^{-\beta} \mod p \\ M &= m^d s^{-\beta} \mod p \\ h &= hash_q(c, G, M) \\ w &= \beta - h \mod q \\ r &= (\alpha - w) x_b^{-1} \mod q. \end{aligned}$$

In the appendix, we show how a third party, Cindy, can act as a middle person to prepare fake signatures under Alice's name intended to Bob. We also give a countermeasure to this problem.

3.2 Ring Signatures

We only present here the restricted case of the ring signature of [11] that implements a designated verifier signature scheme. In [11], two versions are proposed. Here, we only consider the version based on the RSA public-key cryptosystem that is more efficient when the number of potential signers is fixed to two.

We denote by n_a, e_a and n_b, e_b Alice's and Bob's public keys, respectively, and by d_a and d_b the respective private keys.

Signature generation. When Alice wants to sign a message m for Bob, she chooses two random values $v, x_b \in \{0,1\}^c$, where 2^c is larger than both n_a and n_b. Then she computes $x_b^{e_b} \mod n_b$ and extends the result over $\{0,1\}^c$ (for a description of this extension, see [11], section 3.1. In the following, we assume that all the calculations are extended over $\{0,1\}^c$). Now, she solves the following equation in order to determine the value of y:

$$E_{h(m)}\left(x_b^{e_b} \mod n_b \oplus E_{h(m)}(y \oplus v)\right) = v$$

where $h(.)$ is a one-way hash function and $E_k(.)$ denotes a symmetric encryption with k as the key.

Using her private key, Alice goes on computing x_a, such that $x_a^{e_a} \mod n_a = y$. Alice's signature on message m is then $s = (v, x_a, x_b)$.

Signature verification. Anybody can verify the signature by simply checking whether

$$E_{h(m)}\left(x_b^{e_b} \mod n_b \oplus E_{h(m)}\left(x_a^{e_a} \mod n_a \oplus v\right)\right) = v.$$

Transcript simulation. The above signature will not convince Cindy that s is a valid Alice's signature. This is due to the fact that Bob can also compute a signature s', such that the verification succeeds. To do so, Bob chooses $v', x'_a \in \{0,1\}^c$, computes ${x'}_a^{e_a} \mod n_a$ and solves the equation

$$E_{h(m)}\left(y' \oplus E_{h(m)}\left({x'}_a^{e_a} \mod n_a \oplus v'\right)\right) = v'.$$

Now Bob computes x'_b, such that ${x'}_b^{e_b} \mod n_b = y$ and produces $s' = (v', x'_a, x'_b)$ which is undistinguishable from an Alice's signature.

Note that the public-keys e_a and e_b may be set to 3, which allows both exponentiations in the verification phase as well as one of the exponentiations in the signature generation to be replaced by two modular multiplications. So the scheme requires only one modular exponentiation altogether. This, however, is computed with respect to a large RSA exponent, that means that for an equivalent security level, an exponentiation in the RST scheme is more complex than 3 exponentiations in the JSI scheme (see section 7).

4 Description of the Scheme

As is the case in all DL based schemes, we assume that some common parameters are initially shared between the users: a large prime p, a prime factor q of $p-1$, a generator $g \in \mathbb{Z}_p^*$ of order q and a one-way hash function h that outputs values in $\mathbb{Z}_q$.

Each user i chooses his secret key $x_i \in \mathbb{Z}_q$ and publishes the corresponding public key $y_i = g^{x_i} \mod p$.

Signature generation. In order to sign a message m for Bob, Alice selects two random values $k \in \mathbb{Z}_q$ and $t \in \mathbb{Z}_q^*$ and computes

$$\begin{aligned} c &= y_b^k \mod p, \\ r &= h(m, c), \\ s &= kt^{-1} - rx_a \mod q. \end{aligned}$$

The triple (r, s, t) is then the signature of the message m.

Verification. Knowing that a signature is originated from Alice, Bob may verify its validity by checking whether $h(m, (g^s y_a^r)^{tx_b} \mod p) = r$.

As we can see, nobody else other than Bob can perform this verification, since his secret key is involved in the verification equation. Hereafter, we show that even if Bob reveals his secret key, he cannot convince another party, Cindy, of the validity of such a signature.

Indeed when Cindy is given Bob's secret key, she can certainly check the consistency of the signature in the same way as Bob. But, there is no reason that she accepts it as an Alice's signature, because Bob is capable to generate the same transcripts in an indistinguishable way.

Transcript simulation. Bob selects $s' \in \mathbb{Z}_q$ and $r' \in \mathbb{Z}_q^*$ at random and compute

$$\begin{aligned} c &= g^{s'} y_a^{r'} \mod p \\ r &= h(m,c) \\ \ell &= r' r^{-1} \mod q \\ s &= s' \ell^{-1} \mod q \\ t &= \ell x_b^{-1} \mod q. \end{aligned}$$

Then $c = (g^s y_a^r)^{t x_b} \mod p$ and $h(m,c) = r$. In fact

$$\begin{aligned} (g^s y_a^r)^{t x_b} &\mod p = \\ (g^s y_a^r)^{\ell} &\mod p = \\ g^{s\ell} y_a^{r\ell} &\mod p = \\ g^{s'} y_a^{r'} &\mod p = c \end{aligned}$$

and $h(m,c) = r$ by definition.

Remarks:

1. To be complete, let us notice that Cindy should start by checking whether the received secret key is actually the Bob's one ($g^{x_b} \mod p \overset{?}{=} y_b$), because without it she is even not convinced that the signature is made by "Alice or Bob", as anybody may have simulated Alice's signature (intended to himself) and give his secret key to Cindy, claiming that it is the Bob's one.
2. Instead of revealing his secret key, Bob can prove to Cindy the consistency of a signature (and not that it is originated by Alice) as follows. Bob presents (m, s, t, c) to Cindy. Then Cindy computes $r = h(m,c)$ and asks to Bob to prove the knowledge of x_b as the discrete logarithm of y_b in one hand and of c on the other hand with respect to g and $(g^s y_a^r)^t \mod p$ as the bases, respectively.

5 Security

In [9] and [10], Pointcheval and Stern discussed the security of a large class of signature schemes, namely those that are derived from zero-knowledge identification protocols by replacing the verifier's role with some hash functions. They argued that in order to prove such schemes secure, it is essential to consider the hash function in use as a random function and showed how to prove the security using the Random Oracle Model [2] and a new lemma, called the Forking Lemma.

We assume that the reader is familiar with the random oracle model and the forking lemma. We just note that the security proofs in this model are based on the following statement: "If there exists a probabilistic polynomial time Turing machine that outputs with non-negligible probability a valid signature linked to a given user without knowing the related secret key, then a polynomial replay of that machine with the same random tape and a different oracle will produce two valid signatures, leading to the resolution of the hard problem on which the signature scheme relies".

At the first glance, our designated verifier signature scheme seems to satisfy all the necessary properties in order to be analysed in this model. However, this cannot be the case because of the designated verifier's capability of simulating signatures. Indeed in our scheme, Bob can simulate as many Alice's signatures as he desires (with himself as the designated verifier) without being able to derive Alice's secret key from them. What makes this impossible is that if Bob reuses the same random values r' and s', but with different oracles in simulating two signatures, he will only obtain linearly dependent transcripts that are clearly of no use for deriving Alice's secret key:

$$\begin{cases} c = g^{s'} y_a^{r'} \mod p \\ r = h(m, c) \\ \ell = r' r^{-1} \mod q \\ s = s' \ell^{-1} \mod q \\ t = \ell x_b^{-1} \mod q \end{cases} \quad \text{and} \quad \begin{cases} c = g^{s'} y_a^{r'} \mod p \\ \hat{r} = \hat{h}(m, c) \\ \hat{\ell} = r' \hat{r}^{-1} \mod q \\ \hat{s} = s' \hat{\ell}^{-1} \mod q \\ \hat{t} = \hat{\ell} x_b^{-1} \mod q \end{cases}$$

$$\Longrightarrow (g^s y_a^r)^{t x_b} = (g^{\hat{s}} y_a^{\hat{r}})^{\hat{t} x_b}$$
$$\Longrightarrow (g^{s'\ell^{-1}} y_a^{r'\ell^{-1}})^{\ell} = (g^{s'\hat{\ell}^{-1}} y_a^{r'\hat{\ell}^{-1}})^{\hat{\ell}}$$
$$\Longrightarrow g^{s'} y_a^{r'} = g^{s'} y_a^{r'}$$

Hence, since there is no assumption on the behaviour of the attackers in the random oracle model, some attackers may act as Bob and produce signatures such that a replay with the same random tape and a different oracle generates a second signature that is linearly dependent to the first one, i.e., without being able to derive Alice's secret key. This means that our scheme may not be proved secure in the above model.

Fortunately, the security proof in our case is even simpler than for classical signature schemes. In our model, we do not require the possibility of replaying the attacker or using two different oracles; all we need is the assumption that the hash function is considered as an oracle that on each valid input produces a random value and that this input should be known to the attacker beforehand. With this in mind, we prove that

Theorem 1. *If a valid Alice's signature for Bob (as the designated verifier) can be generated without the knowledge of Alice's or Bob's secret keys, then the Diffie-Hellman problem may be solved in polynomial time.*

Proof (sketch). Let T be a probabilistic polynomial time Turing machine that receives y_a, y_b and g as input. We assume that T may ask a polynomial number of questions to the random oracle and that they are of the form (m_i, c_i). We also assume that these questions are stored in a table together with the related answers.

Now suppose that T can find with non-negligible probability a valid signature (t, r, s) on a message m, such that r is the output of the oracle on the query (m, c), for some c that is computed or selected by T beforehand. We consider a variant T' of T that uses the generated signature to compute the Diffie-Hellman shared key of Alice and Bob, i.e., $g^{x_a x_b}$. To do so, T', after having seen the output of T, searches in the table of the stored questions and answers to find c corresponding to m and r. Now, T' can determine $g^{x_a x_b} = (c^{t^{-1}} y_b^{-s})^{r^{-1}}$, because

$$\begin{aligned}(c^{t^{-1}} y_b^{-s})^{r^{-1}} &= (((g^s y_a^r)^{t x_b})^{t^{-1}} y_b^{-s})^{r^{-1}} \\ &= ((g^s y_a^r)^{x_b} y_b^{-s})^{r^{-1}} \\ &= (y_b^s y_b^{r x_a} y_b^{-s})^{r^{-1}} \\ &= (y_b^{r x_a})^{r^{-1}} \\ &= y_b^{x_a} \\ &= g^{x_a x_b}.\end{aligned}$$

Remarks:

3. The interesting question that arises from this proof is the following. If forging a single signature allows to determine $g^{x_a x_b}$, why the knowledge of a "real" signature does not lead to the calculation of this key. The answer is clear: with a real signature (t, r, s), nobody, without the knowledge of the designated verifier's secret key, can compute the related c that is required for the computation of the Diffie-Hellman shared key of two users. However, in order to forge a signature, one has, by assumption, to know this value in order to prepare the input of the oracle.
4. Once the value c related to a signature (t, r, s) on the message m becomes known to an attacker, it is possible for her to forge signatures under Alice's name for Bob on any message $\hat{m}$ of her choice. In fact, it suffices to compute $\hat{r} = h(\hat{m}, c)$ and set $\hat{s} = sd \mod q$ and $\hat{t} = td^{-1} \mod q$, where $d = \hat{r}r^{-1} \mod q$. In this case,

$$\begin{aligned}(g^{\hat{s}} y_a^{\hat{r}})^{\hat{t} x_b} &\mod p = \\ (g^{sd} y_a^{rd})^{td^{-1} x_b} &\mod p = \\ (g^s y_a^r)^{x_b} &\mod p = c\end{aligned}$$

and $\hat{r} = h(\hat{m}, c)$, by construction.
However, since this kind of signature is designed to be verifiable just for a particular verifier, it is only up to him to reveal c to other people and make them capable to fool him in the future.

Theorem 2. *The above signature scheme is designated verifier.*

Proof. We have to show that the transcripts simulated by Bob are indistinguishable from those that he receives from Alice, i.e., the following distributions are identical:

$$\sigma = (r, s, t) : \begin{cases} k \in_R \mathbb{Z}_q \\ t \in_R \mathbb{Z}_q^* \\ r = h(m, y_b^k \mod p), \\ s = kt^{-1} - rx_a \mod q \end{cases}$$

and

$$\sigma' = (r, s, t) : \begin{cases} s' \in_R \mathbb{Z}_q \\ r' \in_R \mathbb{Z}_q^* \\ r = h(m, y_a^{r'} g^{s'} \mod p), \\ s = s'r'^{-1}r \mod q \\ t = r'r^{-1}x_b^{-1} \mod q \end{cases}$$

Let $(\hat{r}, \hat{s}, \hat{t})$ be a signature that is randomly chosen in the set of all valid Alice's signatures intended to Bob. Then we have:

$$\Pr_{\sigma}[(r, s, t) = (\hat{r}, \hat{s}, \hat{t})] = \Pr_{k;t \neq 0} \begin{bmatrix} r = h(m, y_b^k \mod p) = \hat{r} \\ t = \hat{t} \\ s = kt^{-1} - rx_a \mod q = \hat{s} \end{bmatrix} = \frac{1}{q(q-1)}$$

and

$$\Pr_{\sigma'}[(r, s, t) = (\hat{r}, \hat{s}, \hat{t})] = \Pr_{s';r' \neq 0} \begin{bmatrix} r = h(m, y_a^{r'} g^{s'} \mod p) = \hat{r} \\ t = r'r^{-1}x_b^{-1} \mod q = \hat{t} \\ s = s'r'^{-1}r \mod q = \hat{s} \end{bmatrix} = \frac{1}{q(q-1)}$$

that means that both distributions of probabilities are the same.

6 Strong Designated Verifier Signatures and Practical Issues

In [8], it is suggested that, in order to make designated verifier signatures strong, transcripts may be probabilistically encrypted using the public key of the intended verifier. This would guarantee that Cindy, who does not know the intended verifier's secret key, cannot verify such a signature, nor distinguish the transcripts from random strings of the same length and distribution.

This, however, requires the encryption of all or part of the transcripts with the intended verifier's public key, or alternatively the encryption of a session key with which the transcripts should be encrypted using a symmetric cipher, whence an additional complex operation.

As we noticed earlier, the verification of validity or invalidity of signatures in our scheme may only be performed by the designated verifier, since his secret key is involved in the verification. Therefore, all a third party may observe is a set of transcripts that are actually indistinguishable for her from random strings of the same length and distribution. This means that our scheme is inherently strong without any encryption or other operations.

In some circumstances, however, Alice may wish to encrypt the message itself, in order to prevent Cindy to read it (even though she knows that Cindy cannot get convinced of the origin of the message). With strong versions of the two schemes presented in section 3, if an encrypted session key is used for encrypting transcripts, the same session key may be used to encrypt the message. In other words, if the strongness is required, the privacy may be obtained at virtually no cost. In our scheme, since no encryption of transcripts is needed, a session key should still be established between Alice and Bob to encrypt and decrypt the message. But, this is already accomplished during the protocol: since both Alice and Bob can compute some value (namely c) without other parties knowing it, it may serve as a shared session key with which the message can be encrypted. In this case, instead of signing a message for Bob, Alice has to “signcrypt” it as follows.

She selects two random values $k \in \mathbb{Z}_q$ and $t \in \mathbb{Z}_q^*$ and computes $c = y_b^k \mod p$. Then she splits c into c_1 and c_2 of appropriate lengths and computes

- $z = E_{c_1}(m)$
- $r = h(m, c_2)$
- $s = kt^{-1} - rx_a \mod q$

where E is a symmetric encryption algorithm. (r, s, t, z) is then the signcryption of the message m.

Knowing that this signcrypted message is received from Alice, Bob

- computes $c = (g^s y_a^r)^{tx_b} \mod p$,
- splits c into c_1 and c_2 (as Alice does),
- finds $m = D_{k_1}(z)$, where D is the decryption algorithm, and
- verifies whether $h(m, c_2) = r$.

If so, Bob makes sure that m is the original message signcrypted by Alice.

7 Comparison

In this section, we give a performance comparison of our scheme and the two existing ones, namely the JSI and the RST schemes. For this comparison, we choose an implementation, setting $p = 512$ bits and $q = 160$ bits for the JSI and our scheme. In order to have comparable security, we set the RSA modulus to 512 bits in the RST scheme.

Table 1. Performance and size comparison

	JSI	Strong JSI	RST	Strong RST	New scheme	Strong new scheme
Generation	1,200	+0	768	+0	240	+0
Verification	1,200	+768	0	+768	480	+0
Total	2,400	+768	768	+768	720	+0
Size (bits)	2,368	+512	1,536	+512	480	+0

For the comparison to be effective, we only consider the number of modular exponentiations, which are the most time-consuming operations, and neglect other operations such as hashing, modular multiplications and symmetric encryptions. We also suppose that when using RSA, the public-key is set to 3, which allows one exponentiation to be replaced by two modular multiplications that are considered negligible.

In table 1, we indicate the complexity—in terms of modular multiplications resulting from modular exponentiations—of the two existing schemes in their strong and no strong flavours, as well as the new scheme. We assume that an exponentiation is equivalent to $1.5 \times \ell$ modular multiplications, where ℓ is the size of the exponent. In order for the JSI and the RST schemes to provide the strong designated verifier property, they need to be encrypted. We assume that a session key is encrypted using 512 bit RSA public-key encryption. This session key can then be used to cipher the transcripts.

We can see in table 1 that our scheme is much more efficient than the JSI scheme for both generation and verification. One may also see that the verification in the RST scheme is the most efficient. However this is not crucial for designated verifier signatures. In traditional signature schemes, efficient verification is a desirable design issue, motivated by the fact that a signature is generated once, but may be verified many times. In designated verifier schemes, there exists only one verifier, which implies only one verification. Therefore we argue that, for designated verifier schemes, only the total computational cost, regrouping signature generation and verification, is significant. When considering the total computing amount, our scheme is slightly more efficient than the RST scheme in the normal case, and more than twice more efficient in case of strong designated verifier.

Note that, our scheme may also be compared with the 1-out-of-n signature scheme [1]. If we consider, for instance, the discrete-logarithm case of this scheme (that uses the Schnorr signature scheme as the basis), our scheme is at least twice more efficient, even without requiring the strongness.

Finally, we would like to emphasize on the size of the respective signatures. Assuming that the hash function's output is of 160 bits, our scheme provides significantly smaller signatures compared with the two other, namely 3 times less than the RST signatures and 5 times less than those of the JSI.

8 Conclusion

In this paper, we proposed a new designated verifier signature scheme. To the best of our knowledge, it is the first scheme providing directly the strong designated verifier property, without any additional encryption. We introduced new definitions of designated verifier proofs and analysed the security of our scheme based on those definitions. We also compared the new scheme with the existing ones and showed that it is more efficient in terms of both computation and communication complexity, especially when the strongness is required.

References

[1] M. Abe, M. Ohkubo and K. Suzuki, 1-out-of-n signatures from a variety of keys, Advances in Cryptology (Proceedings of Asiacrypt '02) , Lecture Notes in Computer Science, vol. 2501, Springer-Verlag, 2002, pp. 415-432. 44, 52

[2] M. Bellare and P. Rogaway, Random oracles are practical: a paradigm for designing efficient protocols, Proceedings of the 1st CCCS, ACM Press, 1993, pp. 62-73. 48

[3] D. Chaum, Private signature and proof systems, United States Patent 5,493,614, 1996. 40, 44

[4] D. Chaum and H. Van Antwerpen, Undeniable signatures, Advances in Cryptology (Proccedings of Crypto '90), Lecture Notes in Computer Science, vol. 435, Springer-Verlag, 1991, pp. 212-216. 40

[5] Y. Desmedt, C. Goutier and S. Bengio, Special uses and abuses of the Fiat-Shamir passport protocol, Advances in Cryptology (Proccedings of Crypto '87), Lecture Notes in Computer Science, vol. 293, Springer-Verlag, 1988, pp. 21-39. 40

[6] Y. Desmedt and M. Yung, Weaknesses of Undeniable Signature Schemes (Extended Abstract), Advances in Cryptology (Proccedings of Eurocrypt '91), Lecture Notes in Computer Science, vol. 547, Springer-Verlag, 1992, pp. 205-220. 40

[7] M. Jakobsson, Blackmailing using Undeniable Signatures, Advances in Cryptology (Proccedings of Eurocrypt '94), Lecture Notes in Computer Science, vol. 950, Springer-Verlag, 1995, pp. 425-427. 40

[8] M. Jakobsson, K. Sako and R. Impagliazzo, Designated Verifier Proofs and Their Applications, Advances in Cryptology (Proceedings of Eurocrypt '96) , Lecture Notes in Computer Science, vol. 1070, Springer-Verlag, 1996, pp. 143-154. 40, 41, 42, 43, 44, 50

[9] D. Pointcheval and J. Stern, Security proofs for signature schemes, Advances in Cryptology (Proceedings of Eurocrypt '96), Lecture Notes in Computer Science, vol. 1070, Springer-Verlag, 1996, pp. 387-398. 47

[10] D. Pointcheval and J. Stern, Security arguments for digital signatures and Blind signatures, Journal of Cryptology, vol. 13, no. 3, 2000, pp. 361-396. 47

[11] R. Rivest, A. Shamir and Y. Tauman, How to Leak a Secret, Advances in Cryptology (Proceedings of Asiacrypt '01), Lecture Notes in Computer Science, vol. 2248, Springer-Verlag, 2001, pp. 552-565. 41, 44, 45

[12] C. Schnorr, Efficient signature generation by smart cards, Journal of Cryptology, vol. 4, no. 3, 1991, pp. 161-174. 42

[13] Y. Zheng, Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption), Advances in Cryptology (Proccedings of Crypto '97), Lecture Notes in Computer Science, vol. 1294, Springer-Verlag, 1997, pp. 165-179. 42

A Cryptanalysis and Enhancement of the JSI Scheme

The first requirement in a designated verifier signature scheme is that Bob, the intended verifier, should trust on the authenticity and the origin of the signatures he receives. Hereafter, we show that when Alice signs a message m for Bob as the intended recipient, Cindy can transform it to an Alice's signature on a message m' of her choice without Bob being able to detect the forgery.

More precisely, Cindy observes a potential "message,signature" pair (m, s) with its associated proof $\sigma = (w, r, G, M, d)$, and replaces (m, s) with a "chosen-message,fake signature" pair (m', s'), such that the proof is left intact. To do so, Cindy computes $s' = (m'^d M^{-1})^{(h+w)^{-1}} \mod p$. Then Bob, who recieves the pair (m', s') and the proof σ, will obviously accept it as originated from Alice.

The reason that this attack works is because neither m' nor s' are connected in any way to the transcripts of σ, i.e., m' can be freely chosen and s' can be computed accordingly without affecting any part of the proof. So, to overcome this shortcoming, one of these values should be bound to σ, by means of the hash function. That is, if h is computed as $hash_q(c, G, M, s)$, then the new value of s (s' corresponding to the chosen m') would modify h, that is needed to be fixed in advance for computing s'.

Note that without putting m in the hash function, one may still fix s', compute h and derive the corresponding m' as $m' = (Ms'^{h+w})^{d^{-1}} \mod p$, i.e., the scheme would still be subject to existential forgery. However, since the JSI scheme is a non-interactive designated verifier proof of Chaum's undeniable signatures, it is assumed that m is already the result of a hashing, i.e., m =hash(original message). Therefore, once m' is computed as above, the attacker is left with a hard problem to derive the original messages, i.e., inversing the hash function.

Note also that this attack is not effective with the strong version of the JSI scheme assuming that the transcripts are encrypted.

Sound Computational Interpretation of Formal Encryption with Composed Keys

Peeter Laud[1]* and Ricardo Corin[2]**

[1] Tartu University, Liivi 2, Tartu, Estonia
peeter_l@math.ut.ee
[2] University of Twente, P.O.Box 217, 7500AE The Netherlands
corin@cs.utwente.nl

Abstract. The formal and computational views of cryptography have been related by the seminal work of Abadi and Rogaway. In their work, a formal treatment of encryption that uses atomic keys is justified in the computational world. However, many proposed formal approaches allow the use of *composed keys*, where any arbitrary expression can be used as encryption key. In this paper we consider an extension of the formal model presented by Abadi and Rogaway, in which it is allowed to use composed keys in formal encryption. We then provide a computational interpretation for expressions that allow us to establish the computational soundness of formal encryption with composed keys.

1 Introduction

Usually, it is necessary to adopt an abstract view of cryptographic operations (such as message encryption) to make the design and analysis of cryptographic protocols more manageable.

Two different, but still related abstract views of cryptographic operations –the formal and the computational– have developed separately in the last years. In the former, the exchanged messages of the protocol are modelled as formal expressions of a term algebra. The (cryptographic) operations, such as message pairing and encryption, are modelled as term constructors. In this setting, an adversary and its abilities can be modelled in terms of the messages the adversary knows; see for e.g. [11]. Furthermore, the security properties a protocol is supposed to achieve are also modelled formally [9, 19]. On the other hand, in the computational model, messages are considered to be (more realistically) bit-strings, while cryptographic operations are seen as functions over these bit-strings. Here, an adversary is modelled as any efficient algorithm, while the security properties of a cryptographic protocol are defined on terms of the probability of the adversary to perform a succesful attack [13, 5].

Both of the two above models have advantages and disadvantages. On the one hand, the formal model allows to reason about cryptographic protocols more

* Supported by Estonian Science Foundation, grant No. 5279.
** Supported by the LicenseScript project and the Telematica Instituut.

easily and generally. However, such benefits arise from the adoption of fairly strong assumptions (such as freeness of the term algebra, and fixing the adversary model). On the other hand, the computational model, by considering messages as bit-strings and modelling the adversary as any efficient algorithm, provides a more realistic model and thus offers more convincing security guarantees. However, proving protocols correct in the computational model is more difficult and less general than in the formal model.

In the work of Abadi and Rogaway [3], it is shown that if two formal expressions are similar to a formal adversary, then their corresponding computational interpretations, represented as bit-strings in the computational model, are also indistinguishable to any computational adversary. This result comprises a very important step into relating the formal and computational model.

Composed Keys. In [3], formal encryption is modelled by using atomic keys: that is, a formal expression $\{M\}_K$ represents encryption of message M with key K, where M is again a formal expression and K is an atomic symbol representing the cryptographic key. However, considering only atomic keys in encryption is not sufficient, and sometimes we need to be able to allow encryption with *composed keys*, representing non-atomic, constructed keys. In that setting, the formal language would need to be able to consider expressions of the form $\{M\}_N$, where both M and N are expressions. Considering composed keys as possible encryption keys is important due to that, in protocol design, it is fairly common to construct symmetric keys from shared secrets and other exchanged data as part of the protocol run. Examples of this can be found in the work of Gong [14], and, more recently, in a proposed protocol for achieving private authentication [1]. Moreover, many "real-world" cryptographic protocols use composed keys —see, for example SSL 3.0 [12]. Furthermore, in the formal model, some approaches based on constraint solving have been designed with specific support of composed keys [18] (this work was subsequently improved in [10]) and, more recently [4].

This paper defines a computational interpretation $[\![\cdot]\!]$ for the operation $\{M\}_N$. Briefly, the interpretation $[\![\{M\}_N]\!]$ consists of encrypting $[\![M]\!]$ — the interpretation of M with a key obtained by applying the *random oracle* to $[\![N]\!]$. So, the interpretation of $\{M\}_N$ is quite intuitive. On the other hand, this forces us to use the *random oracle model* as the computational model. Using a random oracle seems to be necessary, since otherwise the goodness of keys might be questioned, as well as the independence of different keys.

We also define a relation $\cong$ over formal expressions and, as our main contribution, we show that $M \cong N$ implies the computational indistinguishability of $[\![M]\!]$ and $[\![N]\!]$.

Related Work. The work of Abadi and Rogaway [3] was later extended in Abadi and Jürjens [2] and Laud [15]. In these works, similar soundness results were obtained for richer formal languages, where instead of considering values of formal expressions, it is dealt with *outputs* of programs. However, differently from the formal language presented in this paper, both of these extended languages still treat the encryption operation as using atomic keys.

Micciancio and Warinschi [17] considered the converse of the soundness result (i.e., completeness of the formal language of [3]). In their work, it is shown that a completeness result can be obtained by considering a stronger encryption scheme, namely an authenticated encryption scheme.

Further extensions of the seminal work [3] deal with *encryption cycles* in expressions. For instance, the expression $\{K\}_K$ contains a trivial cycle: key K is immediately encrypted with itself. In the computational model, the security of a traditional encryption scheme can be compromised if an adversary gets hold of a message containing an encryption cycle. Thus, in the original work of Abadi and Rogaway, formal expressions were restricted to be cycle free. However, further work of Black et al. [8] and Laud [16] has shown that, in fact, this discrepancy can be addressed in two different ways: either by considering a new, stronger security definition of the encryption scheme [8], or by strengthening the adversary model of the formal model, such that it can be able to "break" encryption cycles [16].

Recently, Bellare and Kohno [6] have studied the security of cryptosystems against *related-key* attacks and also provided a construction of a secure cryptosystem against a certain kind of such attacks. Related keys are different from composed keys — a related key is something that is constructed from an already existing good key and some non-key data, whereas a composed key is constructed from non-key data only.

Plan of the paper. In Section 2 we present the formal language. Then, in Section 3 we introduce some basic notions of the computational model that are needed in the sequel, and also present an algorithm for translating formal expressions into computational [distributions of] bit-strings. In Section 4, we introduce an equivalence relation $\cong$ over formal expressions. This equivalence relation $\cong$ is elaborated and illustrated with some examples in Section 5. After that, in Section 6 we present the main contribution, a soundness result that relates the formal and computational models. Finally, Section 7 concludes the paper.

2 Expressions and Patterns

Let **Bool** be the set $\{0, 1\}$ and let **Keys** be the set of *formal keys* — this is a fixed, infinite set of symbols. Intuitively, elements of **Keys** represent cryptographic keys. Also, let **Rnd** be the set of *formal random numbers* — again a fixed, infinite set of symbols disjoint from **Keys**. The use of **Rnd** is needed since usually some of the constructors of formal expressions (such as encryption) represent probabilistic operations. This means that if such an operation is executed twice, even with the same arguments, the results will be different. Thus, the elements of **Rnd** are used to keep track which subexpressions of an expression represent the same invocation of that operation and which subexpressions represent different invocations. Our set of formal expressions **Exp** is defined by the following grammar:

$$\begin{array}{rlll} M, N ::= & b & \text{(bit)} & \\ \mid & K & \text{(key)} & \\ \mid & (M, N) & \text{(pair)} & \\ \mid & \{M\}_N^r & \text{(encryption)} & . \end{array}$$

Here, $b \in \mathbf{Bool}$, $K \in \mathbf{Keys}$ and $r \in \mathbf{Rnd}$. Clearly, we can see that composed keys are allowed in the encryption operation. As the labels r are identifiers of invocations of the encryption algorithm, we demand that whenever we consider two expressions $\{M\}_N^r$ and $\{M'\}_{N'}^r$ with the same label r, then also $M = M'$ and $N = N'$.

Even though Abadi and Rogaway [3] did not use formal random numbers, they assumed that each occurrence of the encryption constructor represents a different invocation of the encryption operation. Furthermore, the later work of Abadi and Jürjens [2] considered a richer language in which they also needed to keep track of different invocations. This was done, similarly to the present work, by using formal random numbers.

Let us define some notation related to the structure of formal expressions. The *subexpression relation* $\sqsubseteq$ is the smallest reflexive transitive relation over $\mathbf{Exp}$ containing $M \sqsubseteq (M, N)$, $N \sqsubseteq (M, N)$, $M \sqsubseteq \{M\}_N^r$ and $N \sqsubseteq \{M\}_N^r$ for all $M, N \in \mathbf{Exp}$ and $r \in \mathbf{Rnd}$. For an expression M, we denote:

$$\begin{aligned} \mathit{keys}(M) &:= \{K \in \mathbf{Keys} : K \sqsubseteq M\} \\ \mathit{rns}(M) &:= \{r \in \mathbf{Rnd} : \{N'\}_N^r \sqsubseteq M \text{ for some } N', N \in \mathbf{Exp}\} \\ \mathit{atoms}(M) &:= \mathit{keys}(M) \cup \mathit{rns}(M) \end{aligned}$$

We call the elements of $\mathit{atoms}(M)$ the *atoms* of M.

Intuitively, a *formal pattern* describes what an adversary is able to see when looking at an expression. The elements P, Q of the set of *formal patterns* **Pat** is defined by the following grammar:

$$\begin{array}{rlll} P, Q ::= & b & \text{(bit)} & \\ \mid & K & \text{(key)} & \\ \mid & (P, Q) & \text{(pair)} & \\ \mid & \{P\}_Q^r & \text{(encryption)} & \\ \mid & \Box^r & \text{(undecryptable)} & . \end{array}$$

Here, $\Box^r$ denote ciphertexts that are encrypted with a key that the adversary does not know, and thus can not "see" inside. We use formal random numbers to differentiate between these ciphertexts, and therefore we require that the formal random numbers used at encryptions be different from formal random numbers used at undecryptables. Now, the relation $\sqsubseteq$, as well as the functions *keys*, *rns* and *atoms* are extended to **Pat**. Finally, note that the sets $\mathit{rns}(P)$ and $\mathit{atoms}(P)$ also contain formal random numbers at the undecryptables.

3 Computational Interpretation

In the computational model, an encryption system is a triple of polynomial-time algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ working with bit-strings. Here, $\mathcal{G}$ and $\mathcal{E}$ are probabilistic algorithms while $\mathcal{D}$ is deterministic. The *key generation algorithm* $\mathcal{G}$ takes as input the security parameter n, represented in unary, and returns a new key. The *encryption algorithm* $\mathcal{E}$ takes as input the security parameter, a key and a plaintext and produces a corresponding ciphertext. Since $\mathcal{E}$ is probabilistic, different invocations of $\mathcal{E}$ may return different ciphertexts. Lastly, the *decryption algorithm* $\mathcal{D}$ takes as input the security parameter, a key and a ciphertext and returns the corresponding plaintext.

Let $\mathbf{0}$ be a fixed bit-string. We say that the encryption system $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is *type-0 secure* [3] if, for all probabilistic polynomial-time (PPT) algorithms $\mathcal{A}^{(\cdot),(\cdot)}$ (with interfaces to two oracles), the difference of probabilities

$$\Pr[\mathcal{A}^{\mathcal{E}(1^n,k,\cdot),\mathcal{E}(1^n,k',\cdot)}(1^n) = 1 \, : \, k, k' \leftarrow \mathcal{G}(1^n)] - \Pr[\mathcal{A}^{\mathcal{E}(1^n,k,\mathbf{0}),\mathcal{E}(1^n,k,\mathbf{0})}(1^n) = 1 \, : \, k \leftarrow \mathcal{G}(1^n)]$$

is negligible in n. A function is negligible if its reciprocal grows faster than any polynomial. In [3], Abadi and Rogaway showed that type-0 security is achievable under standard cryptographic assumptions.

Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a type-0 secure encryption system, such that the distribution $\mathcal{G}(1^n)$ is the uniform probability distribution over $\{0,1\}^{\ell(n)}$, where ℓ is a fixed polynomial. Now, being type-0 secure guarantees that the algorithm $\mathcal{E}$ is probabilistic, and thus we denote by $\mathcal{E}^{\mathbf{r}}$ the invocation of $\mathcal{E}$ with random coin-flips $\mathbf{r} \in \{0,1\}^*$. Thus, if we fix $\mathbf{r}$, the algorithm $\mathcal{E}^{\mathbf{r}}$ is now deterministic. In the security definition, we assume the uniform distribution of $\mathbf{r}$.

Now, let x be a bit-string. A *random oracle* R is a machine that, on query $(1^m, x)$, first checks whether it has been queried with the same values before. If this is the case, then it returns the same answer as before. Otherwise, it proceeds as follows. First, the random oracle creates, uniformly and randomly, a bit-string y of length m. Then, the random oracle records the query $(1^m, x)$ together with y, and then finally y is returned. In the *random oracle model* [7], there is a single random oracle in the world, while all other algorithms and machines are allowed to query this oracle. To be able to translate a model in the random oracle world into a real system, the random oracle needs to be replaced with some "random-looking" function h. Thus, there is a leap of faith involved in applying the results proved in the random oracle model to a real system. Nevertheless, we can still be sure that if the real system is insecure, then this must be caused by h not being a good approximation of R.

Now we are ready to give a computational interpretation to expressions and patterns. With each $P \in \mathbf{Pat}$ we associate a family (indexed by the security parameter) of probability distributions over bit-strings. We denote that family by $[\![P]\!]$. Fig. 1 depicts the algorithm sampling the n-th distribution in that family. First, $\textsc{Initialize}(1^n, P)$ is run and then $\textsc{Convert}(1^n, P)$ is invoked.

```
algorithm INITIALIZE($1^n, P$)
    for all $K \in keys(P)$ do $\tau(K) \leftarrow \mathcal{G}(1^n)$
    $\tau_{bb} \leftarrow \mathcal{G}(1^n)$
    for all $r \in rns(P)$ do $\tau(r) \overset{R}{\in} \{0,1\}^*$

algorithm CONVERT($1^n, P$)
    if $P$ is $K \in$ Keys
        return $\langle \tau(K), \text{“key”} \rangle$
    else if $P$ is $b \in$ Bool
        return $\langle b, \text{“bit”} \rangle$
    else if $P$ is $(P_1, P_2)$
        let $x =$ CONVERT($1^n, P_1$)
        let $y =$ CONVERT($1^n, P_2$)
        return $\langle x, y, \text{“pair”} \rangle$
    else if $P$ is $\{P_2\}^r_{P_1}$
        let $x =$ CONVERT($1^n, P_1$)
        let $y =$ CONVERT($1^n, P_2$)
        let $z = \mathcal{E}^{\tau(r)}(1^n, R(1^{\ell(n)}, x), y)$
        return $\langle z, \text{“ciphertext”} \rangle$
    else: $P$ is $\Box^r$
        let $z = \mathcal{E}^{\tau(r)}(1^n, R(1^{\ell(n)}, \tau_{bb}), \mathbf{0})$
        return $\langle z, \text{“ciphertext”} \rangle$
```

Fig. 1. Algorithm sampling $[\![P]\!]$

Note that if $P_1 \neq P_2$ then if we sample $\langle x, y, \text{“pair”} \rangle \leftarrow [\![(P_1, P_2)]\!]_n$, then the probability for $x = y$ is negligible.

In fact, the existence of the random oracle is itself sufficient for the existence of type-0 encryption systems. In particular, we could have fixed the encryption system, for example, to the one given in [8]. However, we would like to use the random oracle as little as possible and thus we have not fixed it.

Two families of probability distributions over bit-strings D and D' are *indistinguishable* (denoted $D \approx D'$) if for all PPT algorithms $\mathcal{A}$, the difference of probabilities

$$\Pr[\mathcal{A}(1^n, x) \,:\, x \leftarrow D_n] - \Pr[\mathcal{A}(1^n, x) \,:\, x \leftarrow D'_n]$$

is negligible in n. In fact, indistinguishability is the computational equivalent of sameness.

4 Equivalence Relation on Pat

We would like to define an equivalence relation $\cong$ over formal expressions (and more generally, over patterns), such that $M \cong N$ implies $[\![M]\!] \approx [\![N]\!]$. Similarly to Abadi and Rogaway, we define a function *pattern* : **Exp** $\longrightarrow$ **Pat** and state

$M \cong N$ iff *pattern*(M) and *pattern*(N) can be obtained from each other by an α-conversion over keys and formal random numbers. Even though we could also define the function *pattern* similarly to [3], that is by giving the entailment relation $\vdash$ (this relation describes which formal expressions the Dolev-Yao attacker may obtain from a given expression) and replacing the undecryptable encryptions in the expressions by the corresponding "blobs" $\Box$, we chose to give a different equivalence definition. The reason for this is that, if we followed Abadi and Rogaway, then we would have to assume that the expressions M and N do not contain *encryption cycles*. With atomic keys only, an encryption cycle in an expression M is a sequence of keys $K_1, \ldots, K_m$, where K_i is encrypted under K_{i+1} (possibly indirectly) for all $i \in \{1, \ldots, m-1\}$ and K_m is encrypted under K_1, all in the expression M. Even though the definition of type-0 security does not say anything about the security of encryption cycles; in systems where the Abadi and Rogaway results can be applied, encryption cycles cannot occur. However, when considering composed keys, the definition of encryption cycles is likely much more contrieved, because the different parts of the same key have to be kept track of. Therefore, we avoid defining the encryption cycles at all, and thus our definition of $\cong$ applies to all expressions.

Let $P, Q \in \mathbf{Pat}$. The operation $\mathsf{box}_Q(P)$ replaces all encryptions of the form $\{\cdot\}^r_Q$ occurring in P with undecryptables. Formally, the operation $\mathsf{box}_Q(P)$ is defined by

$$\begin{aligned}
\mathsf{box}_Q(b) &= b \\
\mathsf{box}_Q(K) &= K \\
\mathsf{box}_Q((P_1, P_2)) &= (\mathsf{box}_Q(P_1), \mathsf{box}_Q(P_2)) \\
\mathsf{box}_Q(\{P_2\}^r_{P_1}) &= \begin{cases} \Box^r, & \text{if } P_1 = Q \\ \{\mathsf{box}_Q(P_2)\}^r_{\mathsf{box}_Q(P_1)}, & \text{if } P_1 \neq Q \end{cases} \\
\mathsf{box}_Q(\Box^r) &= \Box^r \ .
\end{aligned}$$

We are looking for sufficient conditions for $[\![P]\!] \approx [\![\mathsf{box}_Q(P)]\!]$. In particular, we are going to prove that the following is a sufficient condition. Let T_P be the set of all atoms occurring in P, except that if P has subexpressions of the form $\{\cdot\}^r_Q$, then the keys and random numbers inside that Q do not count. The sufficient condition that we are looking for is $\mathit{atoms}(Q) \not\subseteq T_P$. To state this formally, we define the sets $\mathcal{B}_Q(P)$ for all $P \in \mathbf{Pat}$ in the following way:

$$\begin{aligned}
\mathcal{B}_Q(b) &= \emptyset \\
\mathcal{B}_Q(K) &= \{K\} \\
\mathcal{B}_Q((P_1, P_2)) &= \mathcal{B}_Q(P_1) \cup \mathcal{B}_Q(P_2) \\
\mathcal{B}_Q(\{P_2\}^r_{P_1}) &= \begin{cases} \{r\} \cup \mathcal{B}_Q(P_2), & \text{if } P_1 = Q \\ \{r\} \cup \mathcal{B}_Q(P_1) \cup \mathcal{B}_Q(P_2), & \text{if } P_1 \neq Q \end{cases} \\
\mathcal{B}_Q(\Box^r) &= \{r\}
\end{aligned}$$

and set $T_P := \mathcal{B}_Q(P)$.

If the above condition is fulfilled we say that $P \cong \mathsf{box}_Q(P)$. We also say $P \cong Q$ whenever Q can be obtained from P through some α-conversion applied to its formal keys and random numbers. Finally, we extend $\cong$ to an equivalence relation.

Now we are ready to define the function *pattern*. Let $P \in \mathbf{Pat}$. If there exists some $Q \in \mathbf{Pat}$, such that $P \neq \mathsf{box}_Q(P)$ (i.e. Q occurs as an encryption key somewhere in P) and $P \cong \mathsf{box}_Q(P)$ then we put $\mathit{pattern}(P) := \mathit{pattern}(\mathsf{box}_Q(P))$. Otherwise we put $\mathit{pattern}(P) := P$. It is easy to check that *pattern* is well-defined. Furthermore, the function *pattern* is efficiently computable.

5 Examples

Before going to the proof that equivalence implies indistinguishability of interpretations, let us see some examples. We are not going to repeat the examples given by Abadi and Rogaway [3][1]. In our examples we intend to illustrate and clarify what constitutes an "encryption cycle" in an expression and what does not.

– $\{K_1\}^r_{(K_1,K_2)} \cong \Box^r$. This is so since the atom K_2 of the encryption key (K_1, K_2) does not occur anywhere else.
– $\{(K_1, K_2)\}^r_{(K_1,K_2)} \not\cong \Box^r$. This expression is a clear-cut encryption cycle. However, encryption cycles can be more subtle, as the next two examples show.
– $\{(K_2, K_1)\}^r_{(K_1,K_2)} \not\cong \Box^r$.
– $(\{K_1\}^{r_1}_{(K_1,K_2)}, \{K_2\}^{r_2}_{(K_1,K_2)}) \not\cong (\Box^{r_1}, \Box^{r_2})$.
– $\{\{K_2\}^{r_1}_{K_1}\}^r_{\{K_2\}^{r_1}_{K_1}} \not\cong \Box^r$, but $\{\{K_2\}^{r_2}_{K_1}\}^r_{\{K_2\}^{r_1}_{K_1}} \cong \Box^r$. The first example contains an encryption cycle. The second example, however, does not, because the atom r_1 of the key does not occur anywhere else. These two examples show the importance of formal random numbers.
– $(\{K_1\}^r_{(K_1,K_2)}, K_2) \not\cong (\Box^r, K_2)$. Compared to the first example, the addition of K_2 means that now all atoms of the encryption key occur somewhere else.

6 Correctness Proof

Theorem. *Let $P, Q \in \mathbf{Pat}$, such that $\mathit{atoms}(Q) \not\subseteq \mathcal{B}_Q(P)$. Then $[\![P]\!] \approx [\![\mathsf{box}_Q(P)]\!]$.*

Proof. Assume the contrary, i.e. there is an algorithm $\mathcal{A}$ that can distinguish the families of probability distributions $[\![P]\!]$ and $[\![\mathsf{box}_Q(P)]\!]$. Fig. 2 shows an algorithm (first call INITIALIZE and then CONVERT) sampling either $[\![P]\!]$

[1] The reader checking out these examples should keep in mind that:
– [3] uses no formal random numbers; each occurrence of the encryption constructor is assumed to have a different formal random number attached to it;
– in [3], $M \cong N$ does not imply $[\![M]\!] \approx [\![N]\!]$, if M or N contains encryption cycles.

```
algorithm $\textsc{Initialize}(1^n, P, Q)$
    for all $K \in \mathcal{B}_Q(P)$ do $\tau(K) \leftarrow \mathcal{G}(1^n)$
    for all $r$ in the set
        $\mathcal{B}_Q(P)\backslash\big(\{r' \in \textbf{Rnd} \mid \Box^{r'}$ occurs in $P\} \cup \{r' \in \textbf{Rnd} \mid \{\cdot\}_Q^{r'}$ occurs in $P\}\big)$
                do $\tau(r) \overset{R}{\in} \{0,1\}^*$

algorithm $\textsc{Convert}^{f(\cdot),g(\cdot)}(1^n, P, Q)$
    if $\textsc{Convert}^{f(\cdot),g(\cdot)}(1^n, P, Q)$ has been invoked before
        return the value returned previously
    if $P$ is $K \in \textbf{Keys}$
        return $\langle \tau(K)$, "key"$\rangle$
    else if $P$ is $b \in \textbf{Bool}$
        return $\langle b$, "bit"$\rangle$
    else if $P$ is $(P_1, P_2)$
        let $x = \textsc{Convert}^{f,g}(1^n, P_1, Q)$
        let $y = \textsc{Convert}^{f,g}(1^n, P_2, Q)$
        return $\langle x, y$, "pair"$\rangle$
    else if $P$ is $\{P_2\}_{P_1}^r$
        if $P_1 = Q$
            let $y = \textsc{Convert}^{f,g}(1^n, P_2, Q)$
            let $z \leftarrow f(y)$
        else
            let $x = \textsc{Convert}^{f,g}(1^n, P_1, Q)$
            let $y = \textsc{Convert}^{f,g}(1^n, P_2, Q)$
            let $z = \mathcal{E}^{\tau(r)}(1^n, R(1^{\ell(n)}, x), y)$
        return $\langle z$, "ciphertext"$\rangle$
    else: $P$ is $\Box^r$
        let $z \leftarrow g(\mathbf{0})$
        return $\langle z$, "ciphertext"$\rangle$
```

Fig. 2. Algorithm sampling either $[\![P]\!]$ or $[\![\mathsf{box}_Q(P)]\!]$

or $[\![\mathsf{box}_Q(P)]\!]$, depending on the values of the oracles f and g. If f and g are $\mathcal{E}(1^n, k, \cdot)$ and $\mathcal{E}(1^n, k', \cdot)$, then the algorithm in Fig. 2 samples $[\![P]\!]$. If f and g are both $\mathcal{E}(1^n, k, \mathbf{0})$ then this algorithm samples $[\![\mathsf{box}_Q(P)]\!]$. Composing this algorithm with algorithm $\mathcal{A}$ allows us to break the type-0 security of the encryption system.

We have to show that the algorithm $\textsc{Convert}^{f(\cdot),g(\cdot)}$ can complete its job, i.e. it does not have to access τ at a point where it is undefined. If a key K occurs in P but does not belong to $\mathcal{B}_Q(P)$ then this key only occurs as a subexpression of Q, where Q is used as an encryption key in P. But $\textsc{Convert}^{f(\cdot),g(\cdot)}(1^n, Q, Q)$ is never needed in this context, the oracle f is used instead of encrypting with it. Therefore we do not need the value of K there. Similarly, if $r \in \textbf{Rnd}$ occurs in P but not in $\mathcal{B}_Q(P)$ then we would need it only for computing the interpretation of the encryption key Q, which is not necessary to compute at all. If r belongs to the set subtracted from $\mathcal{B}_Q(P)$ in the algorithm INITIALIZE, then $\tau(r)$ is not used, but the oracles f or g generate some random numbers of their own.

We have to argue that the algorithm in Fig. 2 indeed samples the claimed families of distributions. Clearly, if f and g are both $\mathcal{E}(1^n, k, \mathbf{0})$, then CONVERT$^{f(\cdot),g(\cdot)}(1^n, P, Q)$ samples $[\![\mathsf{box}_Q(P)]\!]$. If f is $\mathcal{E}(1^n, k, \cdot)$ and g is $\mathcal{E}(1^n, k', \cdot)$ then we have to show that the key k used by f is indistinguishable from $R(1^{\ell(n)}$, CONVERT$(1^n, Q, Q))$ even when we are given the values of τ on $\mathcal{B}_Q(P)$. The key used by f is independent of all the given values of τ. Also, these values of τ do not uniquely determine CONVERT$^{f,g}(1^n, Q, Q)$ yet, because $atoms(Q) \not\subseteq \mathcal{B}_Q(P)$. Even more, with given values of τ we can guess the value of CONVERT$^{f,g}(1^n, Q, Q)$ only with negligible success probability. Therefore the application of the random oracle to this value gives us a random bit-string that is independent of the given values of τ. The key used by f and the bit-string $R(1^{\ell(n)}$, CONVERT$(1^n, Q, Q))$ are identically distributed, therefore they are indistinguishable under given conditions. Hence we conclude that CONVERT$^{f(\cdot),g(\cdot)}$ $(1^n, P, Q)$ samples $[\![P]\!]$. $\square$

7 Conclusions

In this paper we have considered an extension of the work of Abadi and Rogaway [3]. This extension is mainly constituted by considering the use of composed, non-atomic keys in the encryption operator of the formal language. Briefly, we proceeded as follows: First, we related formal expressions in our language with an equivalence relation $\cong$. By providing an intuitive computational interpretation, and then showing that each time two formal expressions that are equivalent according to $\cong$ are also indistinguishable in the computational world, we have lifted the work of Abadi and Rogaway [3] to the case of composed keys.

As we already mentioned, support for encryption with composed keys is important since many cryptographic protocols use them [14, 1, 12]. Thus, having the soundness result for the case of formal encryption with composed keys provides further faithfulness in the verification results of formal approaches that support composed keys (such as [18, 10, 4].)

While giving the computational interpretation, we needed to use the random oracle. Thus, our approach gives less security guarantees than the original work of Abadi and Rogaway, based on standard security assumptions. However, we believe the use of the random oracle is necessary to guarantee the goodness and independence of the constructed keys. Usage of the random oracle allow us to model the situation in which a user generates keys in a completely secure manner, which is in accordance with the existing definitions in the computational model. However, in some situations (e.g. when considering composed keys), the key generation process may *not* be a so private activity. In this new setting, an adversary might have some knowledge about the randomness used during the key generation. Furthermore, a stronger and *active* adversary may even have some control over the key generation process. We believe it would be interesting to study such a new scenario, where new and proper definitions (and constructions) would be needed.

Acknowledgements

We would like to thank Sandro Etalle and the anonymous reviewers for useful comments.

References

[1] M. Abadi. Private authentication. In *Proceedings of the 2002 Workshop on Privacy Enhancing Technologies*, pages 27–40. Springer-Verlag, 2003. 56, 64

[2] M. Abadi and J. Jurjens. Formal eavesdropping and its computational interpretation. In *Fourth International Symposium on Theoretical Aspects of Computer Software (TACS2001)*, LNCS. Springer-Verlag, 2001. 56, 58

[3] M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Journal of Cryptology*, number 15, pages 103–127. Springer-Verlag, 2000. 56, 57, 58, 59, 61, 62, 64

[4] D. Basin, S. Modersheim, and L. Vigano. Constraint differentiation: A new reduction technique for constraint-based analysis of security protocols. In *Workshop on Security Protocol Verification. CONCUR 2003*, September 2003. 56, 64

[5] M. Bellare. Practice-oriented provable security. In *Information Security, First International Workshop, ISW '97*, LNCS 1396, pages 221–231, 1997. 55

[6] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *EUROCRYPT 2003*, LNCS 2656, pages 491–506, 2003. 57

[7] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993. 59

[8] J. Black, P. Rogaway, and T. Shrimpton. Encryption scheme security in the presence of key-dependent messages. In *Selected Areas in Cryptography — SAC'02*, LNCS. Springer-Verlag, 2002. 57, 60

[9] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990. 55

[10] R. Corin and S. Etalle. An Improved Constraint-based system for the verification of security protocols. *9th Int. Static Analysis Symp. (SAS)*, LNCS 2477:326–341, september 2002. 56, 64

[11] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. 55

[12] A. Freier, P. Karlton, and P. Kocher. The ssl protocol. version 3.0. 56, 64

[13] O. Goldreich. On the foundations of modern cryptography. *Lecture Notes in Computer Science*, 1294:46–74, 1997. 55

[14] L. Gong. Using one-way functions for authentication. *ACM SIGCOMM Computer Communication Review*, 19(5):8–11, 1989. 56, 64

[15] P. Laud. Semantics and program analysis of computationally secure information flow. In *Programming Languages and Systems: 10th European Symposium on Programming, ESOP 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genoa, Italy, April 2-6, 2001*, volume 2028 of *LNCS*, pages 77–91. Springer-Verlag, 2001. 56

[16] P. Laud. Encryption cycles and two views of cryptography. In *NORDSEC 2002 - Proceedings of the 7th Nordic Workshop on Secure IT Systems (Karlstad University Studies 2002:31)*, pages 85–100, 2002. 57

[17] D. Micciancio and B. Warinschi. Completeness theorems for the abadi-rogaway language of encrypted expressions. To appear. 57

[18] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *ACM Conference on Computer and Communication Security*, volume Proc. 2001, pages 166–175. ACM press, 2001. 56, 64

[19] L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998. 55

On the Security of a New Variant of OMAC

Tetsu Iwata and Kaoru Kurosawa

Department of Computer and Information Sciences,
Ibaraki University
4–12–1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan
{iwata,kurosawa}@cis.ibaraki.ac.jp

Abstract. OMAC is a provably secure MAC scheme which NIST currently intends to specify as the modes recommendation. In August 2003, Mitchell proposed a variant of OMAC. We call it OMAC1$''$. In this paper, we prove that OMAC1$''$ is *less secure* than original OMAC. We show a security gap between them. As a result, we obtain a negative answer to Mitchell's open question — OMAC1$''$ is *not* provably secure even if the underlying block cipher is a PRP.

Keywords: Message authentication code, OMAC, provable security, pseudorandom permutation.

1 Introduction

1.1 Background

CBC MAC [6, 7] is a well-known and widely used message authentication code (MAC) based on a block cipher E. We denote the CBC MAC value of a message M by $\mathrm{CBC}_K(M)$, where K is the key of E. While Bellare, Kilian, and Rogaway proved that the CBC MAC is secure for fixed length messages [2], it is *not* secure for variable length messages.

Therefore, several variants of CBC MAC have been proposed which are provably secure for variable length messages: we have EMAC, XCBC, TMAC and OMAC.

EMAC (Encrypted MAC) is obtained by encrypting $\mathrm{CBC}_{K_1}(M)$ by E again with a new key K_2 [3]. That is,

$$\mathrm{EMAC}_{K_1,K_2}(M) = E_{K_2}(\mathrm{CBC}_{K_1}(M)) \ .$$

Petrank and Rackoff proved that EMAC is secure if the message length is a multiple of n, where n is the block length of E [13].

For arbitrary length messages, we can simply append the minimal 10^i to a message M so that the length is a multiple of n. In this method, however, we must append an entire extra block 10^{n-1} if the size of the message is already a multiple of n. This is a "wasting" of one block cipher invocation.

Black and Rogaway next proposed XCBC to solve the above problem [4]. XCBC takes *three* keys: one k-bit key K_1 for E, two n-bit keys K_2 and K_3 (k

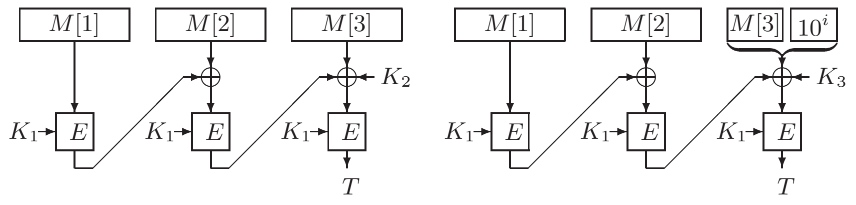


Fig. 1. Illustration of XCBC

denotes the key length of E). In XCBC, we do not append 10^{n-1} if the size of the message is already a multiple of n. Only if this is not the case, we append the minimal 10^i. In order to distinguish them, K_2 or K_3 is XORed before encrypting the last block. XCBC is now described as follows (see Fig. 1).

– If $|M| = mn$ for some $m > 0$, then XCBC computes exactly the same as the CBC MAC, except for XORing an n-bit key K_2 before encrypting the last block.
– Otherwise, 10^i padding ($i = n - |M| - 1 \bmod n$) is appended to M and XCBC computes exactly the same as the CBC MAC for the padded message, except for XORing another n-bit key K_3 before encrypting the last block.

Kurosawa and Iwata then proposed TMAC which requires *two* keys, one k-bit key K_1 and one n-bit key K_2 [10]. TMAC is obtained from XCBC by replacing (K_2, K_3) with $(K_2 \cdot \mathtt{u}, K_2)$, where $\mathtt{u}$ is some non-zero constant and "$\cdot$" denotes multiplication in $\mathrm{GF}(2^n)$. Sung, Hong, and Lee showed a key recovery attack against TMAC [15].

Finally, Iwata and Kurosawa proposed OMAC which requires only *one* block cipher key K [8]. OMAC is a generic name for OMAC1 and OMAC2. Let $L = E_K(0^n)$. Then OMAC1 is obtained from XCBC by replacing (K_1, K_2, K_3) with $(K, L \cdot \mathtt{u}, L \cdot \mathtt{u}^2)$. Similarly, OMAC2 is obtained from XCBC by replacing (K_1, K_2, K_3) with $(K, L \cdot \mathtt{u}, L \cdot \mathtt{u}^{-1})$.

1.2 A New Variant of OMAC1: OMAC1$''$ [12]

EMAC, XCBC, TMAC and OMAC are all provably secure against chosen message attack if the underlying block cipher is a PseudoRandom Permutation (PRP). Indeed, for all of the above MACs, it has been shown that the forging probability is upper bounded by the birthday bound term plus insecurity function of the underlying block cipher as a PRP, which is a standard and acceptable security bound. In fact, many block cipher modes of operations have this security bound. For example we have CTR mode [1] and CBC mode [1] for symmetric encryption, and PMAC [5] for message authentication. Nevertheless, Mitchell proposed a new variant of OMAC1 to *improve* the security of original OMAC1. We call it OMAC1$''$. OMAC1$''$ is obtained from XCBC by replacing (K_1, K_2, K_3) with $(K \oplus S_1, E_K(S_2), E_K(S_3))$, where S_1 is some fixed k-bit constant, S_2 and S_3 are some distinct n-bit constants.

It was claimed that OMAC1$''$ is *more secure* than OMAC1 [12]. Mitchell also posed an open question of whether OMAC1$''$ is provably secure [12].

1.3 Our Contribution

In this paper, however, we show that the security is not improved. We prove that OMAC1$''$ is *less secure* than original OMAC1. We show a security gap between them.

To derive our result we first construct a PRP G with the following property: For any $K \in \{0,1\}^k$,

$$G_K(\cdot) = G_{K \oplus S_1}(\cdot) \ .$$

(A similar PRP is used in [14, 9], and the construction is almost the same as in [9].)

We then show that OMAC1$''$ is completely insecure if G is used as the underlying block cipher.

In particular, we show that there are simple forgery attacks by using only one oracle query. This implies underlying block cipher being a PRP is *not enough* for proving the security of OMAC1$''$. Equivalently, it is *impossible* to prove the security of OMAC1$''$ under the assumption of the underlying block cipher being a PRP. That is,

– OMAC1 is a secure MAC if the underlying block cipher is a PRP [8], while
– it is *impossible* for OMAC1$''$ to achieve this security notion.

Therefore, there is a security gap between OMAC1 and OMAC1$''$, and OMAC1$''$ is *less secure* than OMAC1. This gives a negative answer to Mitchell's open question — OMAC1$''$ is not provably secure even if the underlying block cipher is a PRP.

2 Preliminaries

2.1 Block Ciphers and MACs

Block cipher, E. A block cipher E is a function $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, where, for each $K \in \{0,1\}^k$, $E(K, \cdot)$ is a permutation over $\{0,1\}^n$. We write $E_K(\cdot)$ for $E(K, \cdot)$. k is called the key length and n is called the block length. For TripleDES, $k = 112, 168$ and $n = 64$, and for the AES, $k = 128, 192, 256$ and $n = 128$.

MAC. A MAC is a function $\text{MAC} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$. It takes a key $K \in \{0,1\}^k$ and a message $M \in \{0,1\}^*$ to return an n-bit tag $T \in \{0,1\}^n$. We write $\text{MAC}_K(\cdot)$ for $\text{MAC}(K, \cdot)$. In this paper, we only consider deterministic MACs.

2.2 Security Definitions

Our definitions follow from those given in [11] for PRP, and [2] for the security of MACs.

Security of block ciphers (PRP) [11]. Let Perm(n) denote the set of all permutations on $\{0,1\}^n$. We say that P is a random permutation if P is randomly chosen from Perm(n).

The security of a block cipher $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ as a pseudorandom permutation (PRP) is quantified as $\mathtt{Adv}_E^{\mathsf{prp}}(\mathcal{A})$, the advantage of an adversary $\mathcal{A}$ that tries to distinguish $E_K(\cdot)$ (with a randomly chosen key K) from a random permutation $P(\cdot)$. Let $\mathcal{A}^{E_K(\cdot)}$ denote $\mathcal{A}$ with an oracle which, in response to a query X, returns $E_K(X)$, and let $\mathcal{A}^{P(\cdot)}$ denote $\mathcal{A}$ with an oracle which, in response to a query X, returns $P(X)$. After making queries, $\mathcal{A}$ outputs a bit. Then the advantage is defined as

$$\mathtt{Adv}_E^{\mathsf{prp}}(\mathcal{A}) \stackrel{\text{def}}{=} \left|\Pr(K \stackrel{R}{\leftarrow} \{0,1\}^k : \mathcal{A}^{E_K(\cdot)} = 1) - \Pr(P \stackrel{R}{\leftarrow} \text{Perm}(n) : \mathcal{A}^{P(\cdot)} = 1)\right| \enspace .$$

We say that E is a PRP if $\mathtt{Adv}_E^{\mathsf{prp}}(\mathcal{A})$ is sufficiently small for any $\mathcal{A}$.

Security of MACs [2]. Let MAC : $\{0,1\}^k \times \{0,1\}^* \to \{0,1\}^n$ be a MAC algorithm. Let $\mathcal{A}^{\text{MAC}_K(\cdot)}$ denote $\mathcal{A}$ with an oracle which, in response to a query $M \in \{0,1\}^*$, returns $\text{MAC}_K(M) \in \{0,1\}^n$. We say that an adversary $\mathcal{A}^{\text{MAC}_K(\cdot)}$ *forges* if $\mathcal{A}$ outputs (M, T), where $T = \text{MAC}_K(M)$ and $\mathcal{A}$ never queried M to its oracle $\text{MAC}_K(\cdot)$. We call (M,T) a forgery attempt. Then we define the advantage as

$$\mathtt{Adv}_{\text{MAC}}^{\mathsf{mac}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr(K \stackrel{R}{\leftarrow} \{0,1\}^k : \mathcal{A}^{\text{MAC}_K(\cdot)} \text{ forges}) \enspace .$$

We say that a MAC algorithm is secure if $\mathtt{Adv}_{\text{MAC}}^{\mathsf{mac}}(\mathcal{A})$ is sufficiently small for any $\mathcal{A}$.

2.3 OMAC1 [8]

Let $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. We write OMAC1$[E]$ if we use E as the underlying block cipher in OMAC1. Then OMAC1$[E]$ takes just one k-bit key $K \in \{0,1\}^k$. It takes an arbitrary length message $M \in \{0,1\}^*$ to return an n-bit tag $T \in \{0,1\}^n$.

The algorithm of OMAC1$[E]$ is described in Fig. 2 and illustrated in Fig. 3. In Fig. 2 and Fig. 3,

$$L \cdot \mathtt{u} = \begin{cases} L \ll 1 & \text{if } \mathtt{msb}(L) = 0, \\ (L \ll 1) \oplus \mathtt{Cst}_n & \text{otherwise,} \end{cases}$$

where: (1) $\mathtt{msb}(L)$ denotes the most significant bit of L (meaning the left most bit), (2) $L \ll 1$ denotes the left shift of L by one bit (the most significant

```
**Algorithm** OMAC1[E]_K(M)
$L \leftarrow E_K(0^n)$
$Y[0] \leftarrow 0^n$
Let $M = M[1] \cdots M[m]$, where $|M[i]| = n$ for $i = 1, \ldots, m-1$
**for** $i \leftarrow 1$ **to** $m-1$ **do**
        $X[i] \leftarrow M[i] \oplus Y[i-1]$
        $Y[i] \leftarrow E_K(X[i])$
**if** $|M[m]| = n$ **then** $X[m] \leftarrow M[m] \oplus L \cdot \mathtt{u}$
        **else** $X[m] \leftarrow (M[m]10^{n-1-|M[m]|}) \oplus L \cdot \mathtt{u}^2$
$T \leftarrow E_K(X[m])$
**return** $T$
```

Fig. 2. Definition of OMAC1[E]

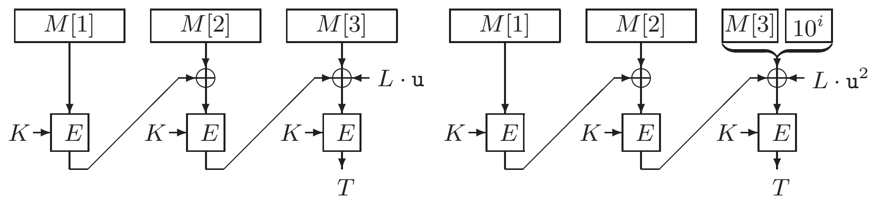


Fig. 3. Illustration of OMAC1[E]

bit disappears and a zero comes into the least significant bit), and (3) $\mathtt{Cst}_n$ is an n-bit constant. For example, $\mathtt{Cst}_{64} = 0^{59}11011$ and $\mathtt{Cst}_{128} = 0^{120}10000111$.

$L \cdot \mathtt{u}^2$ is simply $(L \cdot \mathtt{u}) \cdot \mathtt{u}$. That is,

$$L \cdot \mathtt{u}^2 = \begin{cases} (L \cdot \mathtt{u}) \ll 1 & \text{if } \mathtt{msb}(L \cdot \mathtt{u}) = 0, \\ ((L \cdot \mathtt{u}) \ll 1) \oplus \mathtt{Cst}_n & \text{otherwise.} \end{cases}$$

2.4 A New Variant of OMAC1: OMAC1″ [12]

Mitchell proposed OMAC1″ [12]. Similarly to OMAC1, Let OMAC1″[E] denote OMAC1″, where $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ is used as the underlying block cipher. OMAC1″[E] takes just one k-bit key $K \in \{0,1\}^k$. It takes an arbitrary length message $M \in \{0,1\}^*$ to return an n-bit tag $T \in \{0,1\}^n$.

The algorithm of OMAC1″[E] is described in Fig. 4 and illustrated in Fig. 5.

In Fig. 4 and Fig. 5, S_1 is some fixed k-bit constant, S_2 and S_3 are some distinct n-bit constants.

3 Construction of a PRP, G

In this section, we construct a PRP G with the following property: For any $K \in \{0,1\}^k$,

$$G_K(\cdot) = G_{K \oplus S_1}(\cdot) \ ,$$

```
**Algorithm** OMAC1$''[E]_K(M)$
$L_1 \leftarrow K \oplus S_1$
$L_2 \leftarrow E_K(S_2)$
$L_3 \leftarrow E_K(S_3)$
$Y[0] \leftarrow 0^n$
Let $M = M[1] \cdots M[m]$, where $|M[i]| = n$ for $i = 1, \ldots, m-1$
**for** $i \leftarrow 1$ **to** $m-1$ **do**
    $X[i] \leftarrow M[i] \oplus Y[i-1]$
    $Y[i] \leftarrow E_{L_1}(X[i])$
**if** $|M[m]| = n$ **then** $X[m] \leftarrow M[m] \oplus L_2$
                 **else** $X[m] \leftarrow (M[m]10^{n-1-|M[m]|}) \oplus L_3$
$T \leftarrow E_{L_1}(X[m])$
**return** $T$
```

Fig. 4. Definition of OMAC1$''[E]$

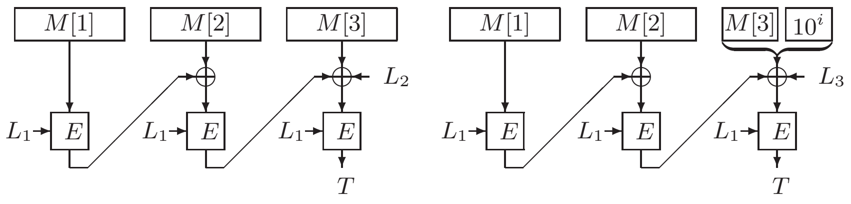


Fig. 5. Illustration of OMAC1$''[E]$. Note that $L_1 = K \oplus S_1$, $L_2 = E_K(S_2)$ and $L_3 = E_K(S_3)$

where S_1 is a non-zero k-bit constant (A similar PRP is used in [14, 9]).

Let $F : \{0,1\}^{k-1} \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a block cipher. It uses a $(k-1)$-bit key K' to encrypt an n-bit plaintext X into an n-bit ciphertext $Y = F_{K'}(X)$, where $F_{K'}(X) \overset{\text{def}}{=} F(K', X)$. For each $K' \in \{0,1\}^{k-1}$, $F_{K'}(\cdot)$ is a permutation over $\{0,1\}^n$.

Now we construct a new block cipher $G : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ from F as in Fig. 6. The inputs to the algorithm are a block cipher F and some non-zero k-bit constant S_1. The output is a new block cipher G.

- For a k-bit string $S_1 = (s_0, s_1, \ldots, s_{k-1})$, $\texttt{nzi}(S_1)$ denotes the smallest index of non-zero element. That is, $\texttt{nzi}(S_1) = j$ such that $s_0 = \cdots = s_{j-1} = 0$ and $s_j = 1$. For example, if $k = 4$ and $S_1 = \texttt{0xA} = \texttt{1010}$, then $\texttt{nzi}(S_1) = 0$, and if $S_1 = \texttt{0x5} = \texttt{0101}$, then $\texttt{nzi}(S_1) = 1$.
- $\texttt{num2str}_{k-1}(i)$ is a $(k-1)$-bit binary representation of i. For example, if $k = 4$ then $\texttt{num2str}_{k-1}(0) = (0,0,0)$ and $\texttt{num2str}_{k-1}(6) = (1,1,0)$.
- For a $(k-1)$-bit string $K' = (K'_0, \ldots, K'_{k-2})$ and an integer $0 \le j \le k-1$, $\texttt{first}_{0..j-1}(K')$ denotes the first j bits of K'. That is, $\texttt{first}_{0..j-1}(K') = (K'_0, \ldots, K'_{j-1})$. For example, if $j = 2$ and $K' = (1,1,0)$ then we have $\texttt{first}_{0..j-1}(K') = (1,1)$, and if $j = 1$ and $K' = (0,1,0)$ then we have $\texttt{first}_{0..j-1}(K') = (0)$. If $j = 0$, then $\texttt{first}_{0..j-1}(K')$ is an empty string.

```
Construction of $G$ from $F$ and $S_1$
$j \leftarrow \mathtt{nzi}(S_1)$;
for $i = 0$ to $2^{k-1} - 1$ do {
        $K' \leftarrow \mathtt{num2str}_{k-1}(i)$;
        $K_1 \leftarrow \mathtt{first}_{0..j-1}(K') \| 0 \| \mathtt{last}_{j..k-2}(K')$;
        $K_2 \leftarrow K_1 \oplus S_1$;
        $G_{K_1} \leftarrow F_{K'}$;
        $G_{K_2} \leftarrow F_{K'}$; }
```

Fig. 6. Construction of G from F and S_1

- Similarly, for a $(k-1)$-bit string $K' = (K'_0, \ldots, K'_{k-2})$ and an integer $0 \le j \le k-1$, $\mathtt{last}_{j..k-2}(K')$ denotes the last $(k-1)-j$ bits of K'. That is, $\mathtt{last}_{j..k-2}(K') = (K'_j, \ldots, K'_{k-2})$. For example, if $j = 2$ and $K' = (1,1,0)$ then $\mathtt{last}_{j..k-2}(K') = (0)$, and if $j = 1$ and $K' = (0,1,0)$ then $\mathtt{last}_{j..k-2}(K') = (1,0)$. If $j = k-1$, then $\mathtt{last}_{j..k-2}(K')$ is an empty string.
- $a \| b$ denotes the concatenation of a and b. For example, if $a = 1$ and $b = (1,0,1)$ then $a \| b = (1,1,0,1)$.

Observe that G_K is well defined for all $K \in \{0,1\}^k$. Indeed, "for loop" in the third line contains 2^{k-1} iterations, and for each loop, two Gs are assigned. Let $K'^{(i)}$, $K_1^{(i)}$ and $K_2^{(i)}$ denote K', K_1 and K_2 in the i-th iteration. Then we see that for any distinct i and i',

- $K_1^{(i)} \neq K_1^{(i')}$ and $K_2^{(i)} \neq K_2^{(i')}$ (since $K'^{(i)} \neq K'^{(i')}$), and
- $K_1^{(i)} \neq K_2^{(i')}$ and $K_2^{(i)} \neq K_1^{(i')}$ (since they differ in the j-th bit).

That is, $K_1^{(i)}$ and $K_2^{(i)}$ in the i-th iteration will not be assigned in the i'-th iteration.

Also observe that we have, for any $K \in \{0,1\}^k$, $G_K(\cdot) = G_{K \oplus S_1}(\cdot)$.

We show two small examples. First, let $k = 4$, $S_1 = \mathtt{0xA} = \mathtt{1010}$ and

$$F = \{F_{000}, F_{001}, F_{010}, F_{011}, F_{100}, F_{101}, F_{110}, F_{111}\},$$

where each $F_{K'}$ is a permutation over $\{0,1\}^n$. In this case, $j = 0$, and for $K' = (K'_0, K'_1, K'_2)$, $K_1 = (0, K'_0, K'_1, K'_2)$, and $K_2 = (1, K'_0, K'_1 \oplus 1, K'_2)$. Then we obtain

$$\begin{aligned} G = \{&G_{0000}, G_{0001}, G_{0010}, G_{0011}, G_{0100}, G_{0101}, G_{0110}, G_{0111},\\ &G_{1000}, G_{1001}, G_{1010}, G_{1011}, G_{1100}, G_{1101}, G_{1110}, G_{1111}\} \end{aligned}$$

where

$$\begin{cases} G_{0000} = F_{000}, G_{0001} = F_{001}, G_{0010} = F_{010}, G_{0011} = F_{011},\\ G_{0100} = F_{100}, G_{0101} = F_{101}, G_{0110} = F_{110}, G_{0111} = F_{111},\\ G_{1000} = F_{010}, G_{1001} = F_{011}, G_{1010} = F_{000}, G_{1011} = F_{001},\\ G_{1100} = F_{110}, G_{1101} = F_{111}, G_{1110} = F_{100}, G_{1111} = F_{101}. \end{cases}$$

```
**Algorithm** $\mathcal{A}^{\mathcal{O}}$
when $\mathcal{B}$ asks its $r$-th query $X_r$:
        return $\mathcal{O}(X_r)$;
when $\mathcal{B}$ halts and output $b$:
        output $b$;
```

Fig. 7. Construction of $\mathcal{A}$

Next, let $k = 4$, and $S_1 = $ `0x5` $=$ `0101`. In this case, $j = 1$, and for $K' = (K'_0, K'_1, K'_2)$, $K_1 = (K'_0, 0, K'_1, K'_2)$, and $K_2 = (K'_0, 1, K'_1, K'_2 \oplus 1)$. Then we obtain

$$\begin{cases} G_{0000} = F_{000}, G_{0001} = F_{001}, G_{0010} = F_{010}, G_{0011} = F_{011}, \\ G_{0100} = F_{001}, G_{0101} = F_{000}, G_{0110} = F_{011}, G_{0111} = F_{010}, \\ G_{1000} = F_{100}, G_{1001} = F_{101}, G_{1010} = F_{110}, G_{1011} = F_{111}, \\ G_{1100} = F_{101}, G_{1101} = F_{100}, G_{1110} = F_{111}, G_{1111} = F_{110}. \end{cases}$$

We note that G can be computed efficiently if F can be computed efficiently. Suppose that we are given a k-bit key K and a plaintext X, and we want to compute $G_K(X)$. Now, let $j \leftarrow \mathtt{nzi}(S_1)$, and check if the j-th bit of K is zero. If it is, let $K' \leftarrow \mathtt{first}_{0..j-1}(K) \| \mathtt{last}_{j+1..k-1}(K)$ and return $F_{K'}(X)$. Otherwise let $K' \leftarrow \mathtt{first}_{0..j-1}(K \oplus S_1) \| \mathtt{last}_{j+1..k-1}(K \oplus S_1)$ and return $F_{K'}(X)$.

We now show that if F is a PRP, then G is a PRP. More precisely, we have the following theorem.

Theorem 3.1. *If* $\mathtt{Adv}_F^{\mathsf{prp}}(\mathcal{A}) \le \epsilon$ *for any adversary* $\mathcal{A}$ *that makes at most* q *queries, then* $\mathtt{Adv}_G^{\mathsf{prp}}(\mathcal{B}) \le \epsilon$ *for any adversary* $\mathcal{B}$ *that makes at most* q *queries.*

Proof. We prove through a contradiction argument. Suppose that there exists an adversary $\mathcal{B}$ such that $\mathtt{Adv}_G^{\mathsf{prp}}(\mathcal{B}) > \epsilon$ where $\mathcal{B}$ asks at most q queries. By using $\mathcal{B}$, we construct an adversary $\mathcal{A}$ such that $\mathtt{Adv}_F^{\mathsf{prp}}(\mathcal{A}) > \epsilon$ where $\mathcal{A}$ asks at most q queries.

The construction is given in Fig. 7. $\mathcal{A}$ has an oracle $\mathcal{O}$ (either P or $F_{K'}$), and $\mathcal{A}$ simply uses $\mathcal{O}$ to answer $\mathcal{B}$'s queries. Finally $\mathcal{A}$ outputs b which is the output of $\mathcal{B}$.

First, suppose that $\mathcal{O} = P$. Then $\mathcal{A}$ gives $\mathcal{B}$ a perfect simulation of a random permutation. Therefore, we have

$$\Pr(P \xleftarrow{R} \mathrm{Perm}(n) : \mathcal{B}^{P(\cdot)} = 1) = \Pr(P \xleftarrow{R} \mathrm{Perm}(n) : \mathcal{A}^{P(\cdot)} = 1) \ .$$

Next, suppose that $\mathcal{O} = F_{K'}$. Then $\mathcal{A}$ gives $\mathcal{B}$ a perfect simulation of G, since from the $\mathcal{B}$'s point of view, each

$$F_{0,\ldots,0}, \ldots, F_{1,\ldots,1}$$

is chosen with probability $1/2^{k-1} = 2/2^k$, which is a precise simulation of G. Note that G is

$$F_{0,\ldots,0}, F_{0,\ldots,0}, \ldots, F_{1,\ldots,1}, F_{1,\ldots,1}$$

```
Algorithm OMAC1''[G]_K(M)
$L_2 \leftarrow G_K(S_2)$
$L_3 \leftarrow G_K(S_3)$
$Y[0] \leftarrow 0^n$
Let $M = M[1] \cdots M[m]$, where $|M[i]| = n$ for $i = 1, \ldots, m-1$
for $i \leftarrow 1$ to $m-1$ do
        $X[i] \leftarrow M[i] \oplus Y[i-1]$
        $Y[i] \leftarrow G_K(X[i])$
if $|M[m]| = n$ then $X[m] \leftarrow M[m] \oplus L_2$
              else $X[m] \leftarrow (M[m]10^{n-1-|M[m]|}) \oplus L_3$
$T \leftarrow G_K(X[m])$
return $T$
```

Fig. 8. Description of OMAC1$''[G]$

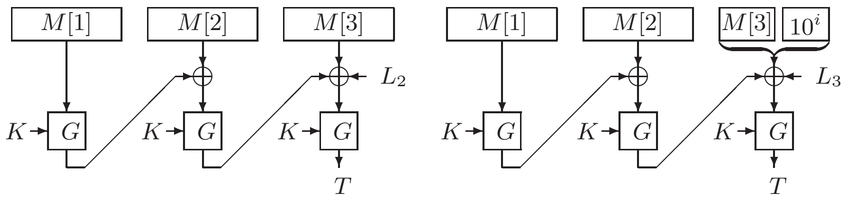


Fig. 9. Illustration of OMAC1$''[G]$. Note that $L_2 = G_K(S_2)$ and $L_3 = G_K(S_3)$

and each $F_{K'}$ is chosen with probability $2/2^k$. Therefore, we have

$$\Pr(K \overset{R}{\leftarrow} \{0,1\}^k : \mathcal{B}^{G_K(\cdot)} = 1) = \Pr(K' \overset{R}{\leftarrow} \{0,1\}^{k-1} : \mathcal{A}^{F_{K'}(\cdot)} = 1) \ .$$

□

4 OMAC1$''[G]$ Is Completely Insecure

Let OMAC1$''[G]$ denote OMAC1$''$, where G is used as the underlying block cipher.

We have the following theorem.

Theorem 4.1. OMAC1$''[G]$ *is not a secure MAC. There exists an adversary* $\mathcal{A}$ *that makes 1 query and* $\mathsf{Adv}^{\mathsf{mac}}_{\mathrm{OMAC1}''[G]}(\mathcal{A}) = 1$.

Proof. Since we have

$$G_K(\cdot) = G_{K \oplus S_1}(\cdot) \ ,$$

the algorithm of OMAC1$''[G]$ can be described as in Fig. 8 and Fig. 9.

We now show two equally efficient attacks against OMAC1$''[G]$.

Attack 1. The adversary first obtains a tag $T \in \{0,1\}^n$ for a two block message $M' = (S_2, S_2) \in \{0,1\}^{2n}$. Then it outputs (M, T), where $M = S_2 \oplus T$, as a forgery attempt.

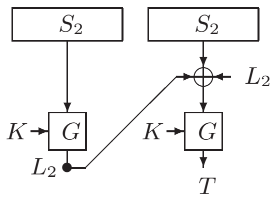


Fig. 10. Illustration of adversary's query. Note that $T = L_2$

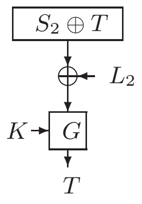


Fig. 11. Illustration of adversary's forgery attempt. We see that $T = \mathrm{OMAC1}''[G]_K(S_2 \oplus T)$

Analysis of Attack 1. For a message $M' = (S_2, S_2) \in \{0,1\}^{2n}$, we have

$$T = \mathrm{OMAC1}''[G]_K(M') = G_K(G_K(S_2) \oplus S_2 \oplus L_2) \ .$$

Since $L_2 = G_K(S_2)$, we have

$$G_K(G_K(S_2) \oplus S_2 \oplus L_2) = G_K(L_2 \oplus S_2 \oplus L_2) = G_K(S_2) = L_2 \ .$$

Therefore, $T = L_2$. See Fig. 10.

Now for a message $M = S_2 \oplus T$ in forgery attempt, we have

$$\mathrm{OMAC1}''[G]_K(M) = \mathrm{OMAC1}''[G]_K(S_2 \oplus T) = G_K(S_2 \oplus T \oplus L_2) \ .$$

Since $T = L_2$, we have

$$G_K(S_2 \oplus T \oplus L_2) = G_K(S_2 \oplus L_2 \oplus L_2) = G_K(S_2) = T \ .$$

See Fig. 11. Therefore, our adversary in attack 1 forges with probability 1.

Attack 2. The adversary first fix some $M' \in \{0,1\}^*$ such that $1 \le |M'| < n$, and then obtains a tag $T \in \{0,1\}^n$ for a two block message $M'' = (S_3, M')$. Then it outputs (M, T), where $M = (M'10^{n-1-|M'|}, S_3 \oplus T, M')$, as a forgery attempt.

Analysis of Attack 2. For a message $M'' = (S_3, M')$, we have

$$T = \mathrm{OMAC1}''[G]_K(M'') = G_K(G_K(S_3) \oplus (M'10^{n-1-|M'|}) \oplus L_3) \ .$$

Since $L_3 = G_K(S_3)$, we have

$$\begin{aligned} G_K(G_K(S_3) \oplus (M'10^{n-1-|M'|}) \oplus L_3) &= G_K(L_3 \oplus (M'10^{n-1-|M'|}) \oplus L_3) \\ &= G_K(M'10^{n-1-|M'|}) \ . \end{aligned}$$

Therefore, $T = G_K(M'10^{n-1-|M'|})$. See Fig. 12.

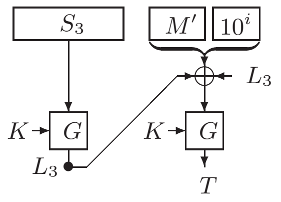


Fig. 12. Illustration of adversary's query. We have $T = G_K(M'10^{n-1-|M'|})$

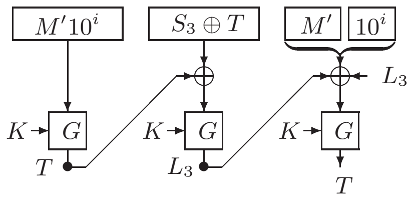


Fig. 13. Illustration of adversary's forgery attempt. We see that $T = \mathrm{OMAC1}''[G]_K(M)$

Now for a message $M = (M'10^{n-1-|M'|}, S_3 \oplus T, M')$ in forgery attempt, we have

$$\begin{aligned}\mathrm{OMAC1}''[G]_K(M) = G_K(G_K(G_K(M'10^{n-1-|M'|}) \oplus S_3 \oplus T)\\ \oplus(M'10^{n-1-|M'|}) \oplus L_3)\ .\end{aligned}$$

Since $T = G_K(M'10^{n-1-|M'|})$, we have

$$\begin{aligned}\mathrm{OMAC1}''[G]_K(M) &= G_K(G_K(T \oplus S_3 \oplus T) \oplus (M'10^{n-1-|M'|}) \oplus L_3)\\ &= G_K(G_K(S_3) \oplus (M'10^{n-1-|M'|}) \oplus L_3)\ .\end{aligned}$$

Since $L_3 = G_K(S_3)$, we have

$$\begin{aligned}\mathrm{OMAC1}''[G]_K(M) &= G_K(L_3 \oplus (M'10^{n-1-|M'|}) \oplus L_3)\\ &= G_K(M'10^{n-1-|M'|})\\ &= T\ .\end{aligned}$$

See Fig. 13. Therefore, our adversary in attack 2 forges with probability 1. □

5 Conclusion

In this paper, we showed that OMAC1″ proposed in [12] are *less secure* than OMAC1. More precisely, we showed that it is *impossible* to prove the security of OMAC1″ under the assumption of the underlying block cipher being a PRP.

References

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. Proceedings of *the 38th Annual Symposium on Foundations of Computer Science, FOCS '97,* pp. 394–405, IEEE, 1997. 68

[2] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *JCSS,* vol. 61, no. 3, 2000. Earlier version in *Advances in Cryptology — CRYPTO '94, LNCS 839,* pp. 341–358, Springer-Verlag, 1994. 67, 70

[3] A. Berendschot, B. den Boer, J. P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgård, M. Dichtl, W. Fumy, M. van der Ham, C. J. A. Jansen, P. Landrock, B. Preneel, G. Roelofsen, P. de Rooij, and J. Vandewalle. Final Report of RACE Integrity Primitives. *LNCS 1007,* Springer-Verlag, 1995. 67

[4] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three key constructions. *Advances in Cryptology — CRYPTO 2000, LNCS 1880,* pp. 197–215, Springer-Verlag, 2000. 67

[5] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology — EUROCRYPT 2002, LNCS 2332,* pp. 384–397, Springer-Verlag, 2002. 68

[6] FIPS 113. Computer data authentication. Federal Information Processing Standards Publication 113, U. S. Department of Commerce / National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1994. 67

[7] ISO/IEC 9797-1. Information technology — security techniques — data integrity mechanism using a cryptographic check function employing a block cipher algorithm. International Organization for Standards, Geneva, Switzerland, 1999. Second edition. 67

[8] T. Iwata and K. Kurosawa. OMAC: One-Key CBC MAC. *Fast Software Encryption, FSE 2003, LNCS 2887,* pp. 129–153, Springer-Verlag, 2003. See http://crypt.cis.ibaraki.ac.jp/. 68, 69, 70

[9] T. Iwata and K. Kurosawa. On the correctness of security proofs for the 3GPP confidentiality and integrity algorithms. *Cryptography and Coding, Ninth IMA International Conference, LNCS 2898,* pp. 306–318, Springer-Verlag, 2003. 69, 72

[10] K. Kurosawa and T. Iwata. TMAC: Two-Key CBC MAC. *Topics in Cryptology — CT-RSA 2003, The Cryptographers' Track at RSA Conference 2003, LNCS 2612,* pp. 33–49, Springer-Verlag, 2003. 68

[11] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.,* vol. 17, no. 2, pp. 373–386, April 1988. 70

[12] C. J. Mitchell. On the security of XCBC, TMAC and OMAC. Technical Report RHUL-MA-2003-4, 19 August, 2003. Available at http://www.rhul.ac.uk/mathematics/techreports. Also available from NIST's web page at http://csrc.nist.gov/CryptoToolkit/modes/comments/. 68, 69, 71, 77

[13] E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *J. Cryptology,* vol. 13, no. 3, pp. 315–338, Springer-Verlag, 2000. 67

[14] P. Rogaway. Comments on NIST's RMAC proposal. Comments to NIST. Available at http://www.cs.ucdavis.edu/~rogaway/xcbc/index.html. Also available at http://csrc.nist.gov/CryptoToolkit/modes/comments/. 69, 72

[15] J. Sung, D. Hong, and S. Lee. Key recovery attacks on the RMAC, TMAC, and IACBC. *The Eighth Australasian Conference on Information Security and Privacy, ACISP 2003, LNCS 2727,* pp. 265–273, Springer-Verlag, 2003. 68

New Methods to Construct Cheating Immune Functions [⋆]

Wen Ping Ma[1] and Moon Ho Lee[2]

[1] National Key Lab. of ISN
Xidian University , Xi'an 710071, P.R.China
wp_ma@hotmail.com

[2] Department of Information and Communication Engineering
Chonbuk National University, Korea
moonho@moak.chonbuk.ac.kr

Abstract. In the paper, the problem of how to construct cheating immune functions is studied. Two new methods to construct balanced quadratic functions on a finite field is proposed, we then present the conditions for this type of balanced quadratic functions can be used in the construction of cheating immune functions. The cryptographic characteristics of these functions are also discussed.

Keywords. Quadratic Function , Balanced Function , MDS Code, Cheating Immune Function

1 Introduction

Secret sharing schemes are widely used in the group oriented cryptography, key management systems, and multiparty secure protocols. But as Tompa and Woll [1] pointed out, many ordinary secret sharing schemes such as the Shamir threshold scheme are subject to cheating by dishonest participants. In fact, all linear secret sharing schemes have this problem. Hence it is very important to study how to prevent cheating by dishonest participants in secret sharing systems.

The concept of cheating immune secret sharing schemes is first proposed by Josef Pieprzyk and Xian-Mo Zhang [2]. They studied the prevention of cheaters and the construction of cheating immune secret sharing schemes in [2, 3, 4, 5]. Cheating immune secret sharing schemes are divided into two classes, ie. the computational secure schemes and unconditional secure ones. In computational secure secret sharing schemes, the combiner checks the validity of the shares submitted by the participants before it reconstructs the shared secret, so any false shares are probably to be found out in this stage and the cheaters are likely to be detected. One solution for computational secure cheating immune

⋆ This work was supported by the Institute of Information Technology Assessment ($IITA$),Chonbuk National University,Korea,and the National Science Foundation of China under the grant No.60373104.

secret sharing is publicly verifiable secret sharing, M.Stadler et. al considered this problem in [6, 7, 8]. Josef Pieprzyk and Xian-Mo Zhang [2] pointed out that cheating by dishonest participants can also be prevented without using the method of public key cryptography. The prevention here is meant that the dishonest participants cannot derive the true shared secret correctly from the invalid secret computed by the combiner, and furthermore the invalid secret reveals no information about the true shared secret.

In [2] Josef Pieprzyk and Xian-Mo Zhang presented some methods to construct cheating immune secret sharing schemes, they also proposed the definition of strict cheating immune functions and their construction methods.

In this paper, we will further study the properties of cheating immune functions. We will present two new methods to construct cheating immune functions,the cryptographic properties of some related quadratic functions are discussed as well.

We organize the rest of this paper as follows. In section 2, we give the basic model of cheating immune secret sharing as in [2]. Section 3 gives our new method to construct cheating immune functions, and section 4 is the conclusion.

2 The Basic Model

The basic model is the same as in [2]. Let $GF(p^t)$ is a finite field with p^t elements, where p is a prime number and t is a positive integer. We use $GF(p^t)^n$ to denote the vector space of dimension n with elements from $GF(p^t)$. Let f is a function from $GF(p^t)^n$ to $GF(p^t)$. Sometimes we may write f as $f(x)$ or $f(x_1, x_2, \cdots, x_n)$, where $x = (x_1, x_2, \cdots, x_n) \in (GF(p^t))^n$, f is said to be balanced if $f(x)$ takes each element of $GF(p^t)^n$ precisely $p^{t(n-1)}$ times. For two vectors $x = (x_1, x_2, \cdots, x_n)$, $\delta = (\delta_1, \delta_2, \cdots, \delta_n)$ in $(GF(p^t))^n$, to define $x_\delta^+ \in (GF(p^t))^n$ and $x_\delta^- \in (GF(p^t))^n$ as the following:

$$(x_\delta^+)_j = \begin{cases} x_j, \text{ if } \delta_j \neq 0 \\ 0, \text{ if } \delta_j = 0 \end{cases}$$

$$(x_\delta^-)_j = \begin{cases} 0, \text{ if } \delta_j \neq 0 \\ x_j, \text{ if } \delta_j = 0 \end{cases}$$

where $j = 1, 2, \cdots, n$.

Let $\tau = (\tau_1, \tau_2, \cdots, \tau_n)$, $\delta = (\delta_1, \delta_2, \cdots, \delta_n)$ be two vectors in $GF(p^t)^n$. We denote $\tau \leq \delta$ if $\tau_i \neq 0$ implies $\delta_i \neq 0$. We use $\tau < \delta$ to denote $\tau \leqq \delta$ and the Hamming weight $HW(\tau)$ of τ (the number of nonzero coordinates of τ) is less than the Hamming weight $HW(\delta)$ of δ. If $\delta^/ \leq \delta$, and $HW(\delta^/) = HW(\delta)$, we write $\delta^/ \approx \delta$. For $\tau, \delta \in (GF(p^t))^n, \delta \neq 0, \tau \leqq \delta$,and $u \in GF(p^t)$, define

$$R_f(\delta, \tau, u) = \{x_\delta^- | f(x_\delta^- + \tau) = u\}.$$

We also simply write $R(\delta, \tau, u)$ in place of $R_f(\delta, \tau, u)$ if there is no confusion by the contexts.

For a secret sharing scheme, let $P = \{P_1, P_2, \cdots, P_n\}$ be the set of the participants, $\alpha = (s_1, s_2, \cdots, s_n) \in (GF(p^t))^n$ be the share vector, ie. s_j is the share distributed to participant P_j by the dealer, $K = f(\alpha)$ be the shared secret. The function f is called the defining function since it determines the secret sharing.

Let $\alpha + \delta$ be the vector consisted of *shares* submitted to the combiner by the participants, we call $\delta = (\delta_1, \delta_2, \cdots, \delta_n) \in (GF(p^t))^n$ the cheating vector, and P_i is a cheater if and only if $\delta_i \neq 0$. The collection of cheaters is determined by the vector $\delta = (\delta_1, \delta_2, \cdots, \delta_n)$ uniquely.

We assume that in pooling phase, dishonest participants always submit invalid shares, and honest participants always submit their valid shares. We also suppose the dishonest participants change their *shares* from time to time, and there is at least one cheater in the system, this implies $\delta \neq 0$.

Consider the vector $\alpha + \delta$, it is obvious that $\alpha + \delta = \alpha_\delta^- + \alpha_\delta^+ + \delta$, here $\alpha_\delta^-$is submitted by the honest participants (or we can say the nonzero coordinates of a_δ^- are shares submitted to the combiner by the honest participants), and $\alpha_\delta^+ + \delta$ by the dishonest ones (the nonzero coordinates of α_δ^+ are shares held by the dishonest participants). In this case, the combiner will output an invalid secret $K^* = f(\alpha + \delta)$.

For the defining function f, share vector α and cheating vector
$\delta = (\delta_1, \delta_2, \cdots, \delta_n)$, the number

$$\rho_{\delta,\alpha} = \frac{\sharp(R(\delta, \alpha_\delta^+ + \delta, K^*) \cap R(\delta, \alpha_\delta^+, K))}{\sharp R(\delta, \alpha_\delta^+ + \delta, K^*)}$$

is the probability of successful cheating by dishonest participants with respect to δ, α , where $\sharp X$ is the cardinality of the set X.

It is obvious that $\rho_{\delta,\alpha} > 0$ since the share vector α is always in the set $(R(\delta, \alpha_\delta^+ + \delta, K^*) \cap R(\delta, \alpha_\delta^+, K))$ and the number of cheaters is equal to $HW(\delta)$. It was proved in [4] that $max\{\rho_{\delta,\alpha} | \alpha \in (GF(p^t))^n\rangle \geq p^{-t}$ for arbitrary $\alpha \in (GF(p^t))^n$ and nonzero $\delta \in (GF(p^t))^n$. If $\rho_{\delta,\alpha} = p^{-t}$ for any $\delta \in (GF(p^t))^n$ with $1 \leq HW(\delta) \leq k < n$ and every $\alpha \in (GF(p^t))^n$, we call f is k -cheating immune.

Let f is quadratic function, if $f(x_\delta^- + \tau + \delta) - f(x_\delta^- + \tau)$ is a non-constant affined function for arbitrary $\delta, \tau \in (GF(p^t))^n$ with $1 \leq (HW(\delta)) \leq k$ and $\tau \leq \delta$, we call f has property $B(k)$.

Theorem 1 ([4]). *Let k,s be two positive integers satisfy $s \geqq (k+1)$, h_i be a balanced quadratic function with property $B(k)$ on $GF(p^t)^{n_i}$ for each $i = 1, 2, \cdots, s$. Set $n = n_1 + n_2 + \cdots + n_s$. Defining the function f on $GF(p^t)^n$ as $f(x) = h_1(y_1) + h_2(y_2) + \cdots + h_s(y_s)$, where $x = (y_1, y_2, \cdots, y_s)$, h_i and h_j have disjoint variables if $i \neq j$. Then the secret sharing with defining function f is k-cheating immune.*

This theorem tells us that cheating immune functions can be constructed from balanced quadratic functions with property $B(k)$.

The following theorem can be proved easily.

Theorem 2. *Let $Q(x_1, x_2, \cdots, x_n) = \sum_{i,j=1,i\leq j}^{n} a_{ij}x_ix_j + \sum_{i=1}^{n} a_ix_i$ be quadratic functions over finite field $GF(q)$,in the case of the finite field is $GF(2)$ or its characteristic is not equal to 2, the function $Q(x_1, x_2, \cdots, x_n)$ is balanced if and only if there exists a $\omega \in GF(q)^n$ such that $Q(x + \omega) - Q(x)$ is a constant and $Q(\omega) \neq 0$.*

3 New Methods to Construct Cheating Immune Functions

The following example demonstrates that not all balanced quadratic functions can be used in the construction of cheating immune functions.

Example 1 : Suppose the characteristic of finite field $GF(q)$ is not equal to 2, $a_1, a_2, a_3, a_4 \in GF(q)$, and $a_1 + a_2 + a_3 + a_4 \neq 0$.

Define the function

$$f(x_1, x_2, x_3, x_4) = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + x_1x_2 - 2x_1x_3 \\ +x_2x_3 - 2x_2x_4 + x_3x_4 + x_1x_4.$$

It is clear that $f(1, 1, 1, 1) = a_1 + a_2 + a_3 + a_4 \neq 0$, and

$$f(x_1 + 1, x_2 + 1, x_3 + 1, x_4 + 1) - f(x_1, x_2, x_3, x_4) = a_1 + a_2 + a_3 + a_4$$

is a constant.

Hence we know from theorem 2 that $f(x_1, x_2, x_3, x_4)$ is balanced.

Now, let $\delta = (\delta_1, 0, \delta_3, 0)$, $\delta_1 \neq 0$, $\delta_3 \neq 0$, $\tau = (b_1, 0, b_3, 0)$, we have

$$f(\delta_1 + b_1, x_2, \delta_3 + b_3, x_4) - f(b_1, x_2, b_3, x_4) = (\delta_1 + \delta_2)x_4 + (\delta_1 + \delta_3)x_2 + c$$

(where c is a constant.) may not be a function of degree 1. Thus $f(x_1, x_2, x_3, x_4)$ does not satisfy the property $B(2)$.

3.1 Scheme1

Let $GF(q)$ be finite field whose characteristic is not equal 2, the generator matrix of a (n, k) maximum distance separate code over $GF(q)$ is A,

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kn} \end{pmatrix}$$

The vector $(a_{(k+1)1}, a_{(k+1)2}, \cdots, a_{(k+1)n})$ is linear independent with all k rows of matrix A,

To construct quadratic function :

$f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} a_{(k+1)i}x_i + \sum_{j=1}^{k}(\sum_{i=1}^{n} a_{ij}x_i)^2$

Theorem 3. *1. The function $f(x_1, x_2, \cdots, x_n)$is balanced.*
2. Suppose $(n-k) \leq k$, the function $f(x_1, x_2, \cdots, x_n)$satisfies the property $B(n-k)$

Proof. 1). To generate vectors $(a_{(k+j)1}, a_{(k+j)2}, \cdots, a_{(k+j)n}), j = 2, \cdots, n-k$, such that $n \times n$ matrix :

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2(n-1)} & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{(n-1)1} & a_{(n-1)2} & \cdots & a_{(n-1)(n-1)} & a_{(n-1)n} \\ a_{n1} & a_{n2} & \cdots & a_{n(n-1)} & a_{nn} \end{pmatrix}$$

is invertible.

To construct linear transform:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

the function $f(x_1, x_2, \cdots, x_n)$ is equivalent to, under the linear transformation :

$$f(y_1, y_2, \cdots, y_n) = y_{(k+1)} + y_1^2 + y_2^2 + \cdots + y_k^2.$$

Obviously, it is balanced. Because the property of balanced is unchanged under linear transformation, thus $f(x_1, x_2, \cdots, x_n)$ is balanced.

2). Let $\delta = (\delta_1, \delta_2, \cdots, \delta_n)$ be cheating vector, $0 < HW(\delta) \leq (n-k)$, Suppose $\delta = (0, \cdots, \delta_{l_1}, \cdots, \delta_{l_m}, \cdots, 0)$, $\tau = (0, \cdots, \tau_{l_1}, \cdots, \tau_{l_m}, \cdots, 0)$ then

$$\begin{aligned} & f(x_{\delta}^- + \tau + \delta) - f(x_{\delta}^- + \tau) \\ = & \sum_{s=1}^{m} a_{(k+1)l_s}\delta_{l_s} + \sum_{j=1}^{k}(\sum_{s=1}^{m} a_{jl_s}\delta_{l_s})(\sum_{s=1}^{m} a_{jl_s}(\delta_{l_s} + 2\tau_{l_s}) + 2\sum_{f \neq l_i} a_{jf}x_f) \end{aligned}$$

Due to the property of matrix A, every $m(m \leq k)$ columns of matrix A is linear independent, thus

$$\sum_{j=1}^{m} a_{jl_s}\delta_{l_s}, j = 1, 2, \cdots, k,$$

are not all zero.

$$\sum_{f \neq l_s} a_{jf}x_f, j = 1, 2, \cdots, k$$

The rank of the coefficient matrix of the equation class above is k, thus

$$f(x_{\delta}^- + \tau + \delta) - f(x_{\delta}^- + \tau)$$

is non-constant affine function, $f(x_2, x_2, \cdots, x_n)$ satisfies the property $B(n-k)$.

3.2 Scheme2

In the following, we are going to give another new method to construct cheating immune defining functions for $(2, n)$ threshold secret sharing schemes. The main idea here is to construct the required cheating immune secret sharing systems from balanced quadratic functions satisfying the property $B(2)$.

First, we construct a $n \times n$ matrix $A(2, n)$ over finite field $GF(p)$, where the characteristic of $GF(p)$ is not equal to 2.

$$A(2,n) = \begin{pmatrix} 0 & 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & -1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & -1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 \\ -1 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & -1 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

Let $a_1, a_2, \cdots, a_n \in GF(p)$ satisfying $\sum_{i=1}^{n} a_i \neq 0$. We now construct the quadratic function

$$f(x_1, x_2, \cdots, x_n) = (x_1, x_2, \cdots, x_n)A(2, n)(x_1, x_2, \cdots, x_n)^T + \sum_{i=1}^{n} a_i x_i$$

Clearly $f(1, 1, \cdots, 1) = \sum_{i=1}^{n} a_i \neq 0$,and
$f(x_1 + 1, x_2 + 1, \cdots, x_n + 1) - f(x_1, x_2, \cdots, x_n) = f(1, 1, \cdots, 1) = constant.$
Hence,due to theorem 2 , the function $f(x_1, x_2, \cdots, x_n)$ is balanced.
Next we discuss when this function satisfies the property $B(2)$
For convenience, let

$$F^{/}(x_1, x_2, \cdots, x_n) = (x_1, x_2, \cdots, x_n)A(2, n)(x_1, x_2, \cdots, x_n)^T$$

and we write $f^{/}(x_0, x_1, \cdots, x_{n-1})$ in stead of $f'(x_1, x_2, \cdots, x_n)$.

If the term $x_i x_j$ occurs in the quadratic function $f'(x_0, x_1, \cdots, x_{n-1})$, we call x_i and x_j are related , in this case, we also call i and j are related. From the construction of f , we know that $(i-1)\ mod\ n, (i-2)\ mod\ n, (i+1)\ mod\ n, (i+2)\ mod\ n$ are related to i for each $i \in \{0, 1, 2, \cdots, n-1\}$. It is clear that $f'(x_0, x_1, \cdots, x_{n-1})$ satisfies the property $B(2)$ if the set $\{(i-1)\ mod\ n, (i-2)\ mod\ n, (i+1)\ mod\ n, (i+2)\ mod\ n\}$ is different from the set $\{(j-1)\ mod\ n, (j-2)\ mod\ n, (j+1)\ mod\ n, (j+2)\ mod\ n\}$ for $i \neq j$.

Theorem 4. *If $n \geq 7$, the set*
$\{(i-1)\ mod\ n, (i-2)\ mod\ n, (i+1)\ mod\ n, (i+2)\ mod\ n\}$
is different from the set
$\{(j-1)\ mod\ n, (j-2)\ mod\ n, (j+1)\ mod\ n, (j+2)\ mod\ n\}$
where $0 \leq i, j \leq (n-1)$, and $i \neq j$.

Proof. For $i \neq j$,if the two set is the same, one of the three situations may occur.

1) $j+1=(i+2)modn, j+2=(i+3)modn, j-1=imodn, j-2=(i-1)modn$
2) $j+1=(i-1)modn, j+2=imodn, j-1=(i-3)modn, j-2=(i-4)modn.$
3) $j+1=(i-2)modn, j+2=(i-1)modn, j-1=(i-4)modn, j-2=(i-5)modn.$

If $n=6$, then situation 3) occurs when $i=0$ and $j=3$. But when $n \geq 7$, none of the three situations would occur.

Theorem 5. *The function $f(x_1, x_2, \cdots, x_n)$ constructed above satisfies the property $B(2)$ if $n \geqq 7$.*

Example 2 : For $n=5$, let

$$f'(x_1, x_2, x_3, x_4, x_5) = x_1(x_2-x_3)+x_2(x_3-x_4)+x_3(x_4-x_5) \\ +x_4(x_5-x_1)+x_5(x_1-x_2).$$

Let $\delta=(0, \delta_2, 0, \delta_4, 0), \delta_2 \neq 0, \delta_4 \neq 0, \tau=(0, b_2, 0, b_4, 0)$. then

$$f^{/}(x_1, \delta_2+b_2, x_3, \delta_4+b_4, x_5)-f(x_1, b_2, x_3, b_4, x_5)$$

may not be an affine function, hence

$$f^{/}(x_1, x_2, x_3, x_4, x_5)$$

does not satisfy the property $B(2)$.

In fact, in example 1, the quadratic terms of the function

$$f(x_1, x_2, x_3, x_4)$$

is equal to

$$(x_1, x_2, x_3, x_4)A(2,4)(x_1, x_2, x_3, x_4)^T.$$

That is the case when $n=4$ in the above construction, thus

$$f(x_1, x_2, x_3, x_4)$$

does not satisfy the property $B(2)$.

From the analysis above, we know neither

$$f(x_1, x_2, x_3)$$

nor

$$f(x_1, x_2, x_3, x_4, x_5, x_6)$$

satisfy the property $B(2)$.

Theorem 6. *The function*

$$f\left(x_1, x_2, \cdots, x_n\right)$$

constructed above satisfies the property $B(m)$ if $n > 3m$.

Proof. Let

$$1 \leqq i_1, i_2, \cdots, i_m \leqq n.$$

For $1 \leqq j \leqq m$, each index in the set $\{\{i_j - 1)modn, (i_j - 2)modn, (i_j + 1)modn, (i_j + 2)modn\}$ is related to i_j When $2n > max\{5m, 4m + 2m\} = 6m$, or $n > 3m$, there is at least one of the numbers

$$(i_k - 1)modn, (i_k - 2)modn, (i_k + 1)modn, (i_k + 2)modn$$

occurs in one and only in one of the set $\{(i_j - 1)modn, (i_j - 2)modn, (i_j + 1)modn, (i_j + 2)modn\}$, $1 \leqq i_1, i_2, \cdots, i_m \leqq n$. This shows that $f(x_1, x_2, \cdots, x_n)$ satisfies the property $B(m)$, Clearly, $f(x_1, x_2, \cdots, x_n)$ is balanced.

4 Conclusion

We Have Studied the Construction of Cheating immune secret sharing functions. New methods to construct cheating immune functions have been presented. Some cryptographic properties of a type of quadratic functions on finite field have also been discussed. The results we have gotten will have significance in the study of cheating immune functions. The methods can be used to construct strictly cheating immune secret sharing and multisecret sharing immune against cheating.

References

[1] M.Tompa and H.Woll, íHow to Share a Secret with cheaters,í Journal of Cryptoligy, Vol.1,No.2 (1988), pp.133-138. 79

[2] Josef Pieprzyk, and Xian-Mo Zhang, Cheating Prevention in Secret sharing on , INDOCRYPT 2001,Lecture Notes in Computer Science, 2247(2001), pp. 226-243, Springer-Verlag. 79, 80

[3] Josef Pieprzyk and Xian-Mo Zhang, Multisecret Sharing Immune against cheating, 2002 international workshop on cryptology and network security. 79

[4] Josef Pieprzyk, Xian-Mo Zhang, Construction of cheating immune secret sharing. ICISC2001,Lecture Note in Computer Science, 2288(2002) ,pp. 226-243, Springer-Verlag. 79, 81

[5] Hossein Ghodosi, Josef Piepreyk, Cheating prevention in secret sharing, ACISP2000, Lecture Notes in Computer Science, 1841(2000), pp. 328-341, Springer-Verlag. 79

[6] Stadler M, Publicly verifiable secret sharing, In Advances in cryptology[A], EUROCRYPTí96[C], LNCS 1070, pp.190-199, Springer-Verlag, Berlin, 1996. 80

[7] Fujisaki E, Okamoto T. A practical and provably secure scheme for publicly verifiable secret sharing and its applications[A], Advances in Cryptology, EUCRYPTOí98[C], pp.32-47, Springer-Verlag, Berlin, 1998. 80

[8] Schoenmakers B. Aíísimple publicly verifiably secret sharing scheme and its application to electronic voting[A], CRYPTOí99[C], ,pp.148-164, Springer-Verlag, Berlin, 1999. 80

Yet Another Definition of Weak Collision Resistance and Its Analysis

Shoichi Hirose

Graduate School of Informatics
Kyoto University, Kyoto, 606-8501 Japan
hirose@i.kyoto-u.ac.jp

Abstract. In this article, a new definition of weak collision resistance is presented following the definition of weak one-wayness. Collision resistance is a security notion of cryptographic hash functions. A collision of a hash function is a pair of different inputs which give the same output. In the new definition, weak collision resistance means that the probability of failing to find a collision is not negligible. This weak collision resistance is then analyzed. First, it is shown that there really exist weakly collision resistant hash functions if collision resistant ones exist. Second, it is shown that weak collision resistance can be amplifiable, that is, (strongly) collision resistant hash functions are constructed from a certain kind of weakly collision resistant ones. An example of weakly collision resistant hash functions is also presented to which the proposed amplification method is applicable.

1 Introduction

Hash functions are very important primitives in cryptography. Hash functions in cryptography are classified in two types: unkeyed hash functions and keyed hash functions. The former ones are also called manipulation detection codes (MDCs). The latter ones are also called message authentication codes (MACs). Excellent surveys are presented in [5, 6].

There are three security notions for cryptographic unkeyed hash functions: preimage resistance, second-preimage resistance and collision resistance. Informally, preimage resistance means that, given an output, it is infeasible to obtain an input which produces the output. Second-preimage resistance means that, given an input, it is infeasible to obtain another input which produces the same output as the given input. Collision resistance means that it is infeasible to obtain two different inputs which produce the same output.

The term "weak collision resistance" is also found in previous literature, and is used to represent two different security notions of hash functions. In some literature, second-preimage resistance is called weak collision resistance [5]. In this case, collision resistance is called strong collision resistance. On the other hand, in [2], weak collision resistance is defined to be collision resistance of keyed hash functions, and it is shown to be implied by unforgeability, that is, secure MAC [1]. In both cases, weak collision rsistance is implied by one-wayness. Hence, from Simon's result [7], it is quite different from (strong) collision resistance. In [7], he showed that no provable construction of a collision resistant hash function exists based on a "black box" one-way permutation, which means that the one-way permutation is used as a subroutine, that is, the internal structure is not used.

For one-wayness, on the other hand, "weak" and "strong" represent the probability of success in finding a preimage. A function is called strongly one-way if, for every probabilistic polynomial time algorithm, the probability that it succeeds in finding a preimage is negligible. A function is called weakly one-way if, for every probabilistic polynomial time algorithm, the probability that it fails to find a preimage is not negligible. In this framework, it is shown that there exist weakly one-way functions if and only if there exist strongly one-way functions [4].

In this article, another kind of definition of weak collision resistance is given following the definition of weak one-wayness. That is, weak collision resistance means that the probability of failure to find a collision is not negligible. In contrast, strong collision resistance means that the probability of success in finding a collision is negligible. The weak collision resistance is then analyzed. First, it is shown that this new definition is not void. It is shown that there exist weakly collision resistant hash functions if there exist collision resistant hash functions. Second, it is shown that weak collision resistance can be amplifiable, that is, strongly collision resistant hash functions can be constructed from weakly collision resistant hash functions satisfying the following property: $h : X \times X \to Y$, $|X| = |Y|$, and both $h(x, \cdot)$ and $h(\cdot, x)$ are permutations for every $x \in X$. These results are obtained in the similar way as in [4]. Furthermore, weakly collision resistant hash functions satisfying the property described above are constructed from collision resistant ones based on discrete logarithms [3].

The rest of this article is organized as follows. In Section 2, weak collision resistance and strong collision resistance are formally defined. In Section 3, the existence of weakly collision resistant hash functions is discussed. The topic of Section 4 is the amplifiability of weak collision resistance. In Section 5, an example of weakly collision resistant hash functions are presented based on discrete logarithms. A concluding remark is in Section 6.

2 Preliminaries

Let $\mathbb{N}$ be the set of positive integers. Let H_n be a set of hash functions such that $H_n = \{h_k \,|\, h_k : D \to R,\ k \in K\}$, where $D \subseteq \{0,1\}^{\ell_D(n)}$, $R \subseteq \{0,1\}^{\ell_R(n)}$, $K \subseteq \{0,1\}^{\ell_K(n)}$, $\ell_D(n) > \ell_R(n)$, and $\ell_D(n)$, $\ell_R(n)$ and $\ell_K(n)$ are polynomials in n. k is regarded as an index. A pair of inputs $(x, x') \in D \times D$ is called a collision of a hash function h_k if $h_k(x) = h_k(x')$ and $x \neq x'$.

A family of hash functions $\{H_n\}_{n \in \mathbb{N}}$ is called weakly collision resistant (weakly CR) if the probability of failure to find a collision is not negligible for every efficient algorithm.

Definition 1. *$\{H_n\}_{n\in\mathbb{N}}$ is called a weakly CR family of hash functions if*

- *there exists a probabilistic polynomial time algorithm K_H which, with an input 1^n, outputs $k \in K$,*
- *there exists a deterministic polynomial time algorithm M_H which, with inputs $k \in K$ and $x \in D$, outputs $h_k(x)$,*
- *and, there exists some polynomial $p(n)$ such that, for every probabilistic polynomial time algorithm F_H and every sufficiently large n,*

$$\sum_{k\in K} \Pr[\mathsf{K}_H(1^n) = k]\ \Pr[\mathsf{F}_H(k) \textit{ fails to find a collision}] \geq \frac{1}{p(n)},$$

where the probability is taken over the coin tosses of K_H and F_H.

A family of hash functions $\{H_n\}_{n\in\mathbb{N}}$ is called strongly CR if the probability of success in finding a collision is negligible for every efficient algorithms.

Definition 2. *$\{H_n\}_{n\in\mathbb{N}}$ is called a strongly CR family of hash functions if*

- *there exist a probabilistic polynomial time algorithm K_H and a deterministic polynomial time algorithm M_H as stated in Definition 1,*
- *and, for every polynomial $q(n)$, every probabilistic polynomial time algorithm F_H and every sufficiently large n,*

$$\sum_{k\in K} \Pr[\mathsf{K}_H(1^n) = k]\ \Pr[\mathsf{F}_H(k) \textit{ succeeds in finding a collision}] < \frac{1}{q(n)},$$

where the probability is taken over the coin tosses of K_H and F_H.

From the definitions, it is obvious that strong CR implies weak CR. In the followings, a family of hash functions is simply called CR if it is strongly CR or weakly CR.

3 Existence of A Weakly Collision Resistant Family of Hash Functions

In this section, it is shown that there exists a weakly CR family of hash functions if there exists a CR family of hash functions.

Let $H_n = \{h_k \mid h_k : D \to R,\ k \in K\}$. For a family of hash functions $\{H_n\}_{n\in\mathbb{N}}$, a family of hash functions $\{F_n\}_{n\in\mathbb{N}}$ is defined as follows.

- $F_n = H_n \cup \{f\}$, where $f : D \to R$, f is polynomial-time computable, and its collision is easy to find.
- K_F is an algorithm for sampling an index. With an input 1^n, it proceeds as follows.
  1. It selects $u_1 \in \{0,1\}^{\lceil \log n \rceil}$ at random.
  2. It runs $\mathsf{K}_H(1^n)$.
  3. It outputs $u = (u_1, u_2)$, where u_2 is the output of $\mathsf{K}_H(1^n)$ in the previous step.

– Let $F_n = \{f_u \mid f_u : D \to R, u \in \{0,1\}^{\lceil \log n \rceil} \times K\}$. Then,

$$f_{(u_1,u_2)} = \begin{cases} h_{u_2} & \text{if } u_1 = 0^{\lceil \log n \rceil} \\ f & \text{otherwise.} \end{cases}$$

It is obvious that $\{F_n\}_{n\in\mathbb{N}}$ is not strongly CR. There exists a probabilistic polynomial time algorithm such that the probability of its success in finding a collision is at least $1 - 1/n$.

Theorem 1. *If $\{H_n\}_{n\in\mathbb{N}}$ is CR, then $\{F_n\}_{n\in\mathbb{N}}$ is weakly CR.*

Proof. Suppose that $\{F_n\}_{n\in\mathbb{N}}$ is not weakly CR. Then, for every polynomial $p(n)$, there exists a probabilistic polynomial time algorithm A_F such that,

$$\sum_{u\in\{0,1\}^{\lceil \log n \rceil}\times K} \Pr[\mathsf{K}_F(1^n) = u] \Pr[\mathsf{A}_F(u) \text{ succeeds}] > 1 - \frac{1}{p(n)}$$

for infinitely many n's. "$\mathsf{A}_F(u)$ succeeds" means that $\mathsf{A}_F(u)$ succeeds in finding a collision, that is, $\mathsf{A}_F(u) = (x, x')$, $f_u(x) = f_u(x')$ and $x \neq x'$. For simplicity, let $m = \lceil \log n \rceil$ and $L = \{0,1\}^m$.

$$\begin{aligned}
&\sum_{u\in L\times K} \Pr[\mathsf{K}_F(1^n) = u] \Pr[\mathsf{A}_F(u) \text{ succeeds}] \\
&= \sum_{\substack{u\in L\times K \\ u_1 = 0^m}} \Pr[\mathsf{K}_F(1^n) = u] \Pr[\mathsf{A}_F(u) \text{ succeeds}] + \\
&\qquad\qquad \sum_{\substack{u\in L\times K \\ u_1 \neq 0^m}} \Pr[\mathsf{K}_F(1^n) = u] \Pr[\mathsf{A}_F(u) \text{ succeeds}] \\
&\le \sum_{u_2\in K} \Pr[\mathsf{K}_F(1^n) = (0^m, u_2)] \Pr[\mathsf{A}_F(0^m, u_2) \text{ succeeds}] + \sum_{\substack{u\in L\times K \\ u_1 \neq 0^m}} \Pr[\mathsf{K}_F(1^n) = u] \\
&= \sum_{u_2\in K} \Pr[\mathsf{K}_F(1^n) = (0^m, u_2)] \Pr[\mathsf{A}_F(0^m, u_2) \text{ succeeds}] + \left(1 - \frac{1}{2^m}\right).
\end{aligned}$$

Thus,

$$\sum_{u_2\in K} \Pr[\mathsf{K}_F(1^n) = (0^m, u_2)] \Pr[\mathsf{A}_F(0^m, u_2) \text{ succeeds}] > \frac{1}{2^m} - \frac{1}{p(n)}.$$

Let A_H be an algorithm which, with an input $k \in K$, runs $\mathsf{A}_F(0^m, k)$ and outputs its output. Then,

$$\begin{aligned}
&\sum_{k\in K} \Pr[\mathsf{K}_H(1^n) = k] \Pr[\mathsf{A}_H(k) \text{ succeeds}] \\
&= 2^m \sum_{u_2\in K} \Pr[\mathsf{K}_F(1^n) = (0^m, u_2)] \Pr[\mathsf{A}_F(0^m, u_2) \text{ succeeds}] \\
&> 1 - \frac{2^m}{p(n)} > 1 - \frac{2n}{p(n)},
\end{aligned}$$

which implies that $\{H_n\}_{n\in\mathbb{N}}$ is not weakly CR. □

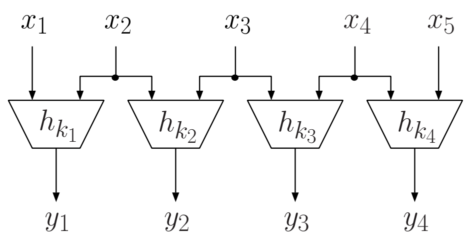


Fig. 1. A Hash Function $g_v \in G_{n,4}$

From the discussion of this section, if there exists a CR family of hash functions, then there exists a weakly CR family of hash functions which is not strongly CR.

4 Amplifiability of Weak Collision Resistance

In this section, it is shown that weak CR can be amplifiable, that is, a strongly CR family of hash functions can be constructed from a weakly CR family of hash functions. Unfortunately, the proposed method of construction is applicable only to the families of hash functions with an additional property. However, in the next section, it is mentioned that there really exists a (weakly) CR family of hash functions with the property.

Theorem 2. *A strongly CR family of hash functions is able to be constructed from any weakly CR family of hash functions*

$$H_n = \{h_k \mid h_k : X \times X \to Y,\ k \in K\}$$

such that $|X| = |Y|$ and both $h_k(x, \cdot)$ and $h_k(\cdot, x)$ are permutations for every k and x.

Proof. Let

$$G_{n,d} = \{g_v \mid g_v : X^{d+1} \to Y^d,\ v \in K^d\}$$

such that $g_v(x_1, x_2, \ldots, x_{d+1}) = (h_{k_1}(x_1, x_2), h_{k_2}(x_2, x_3), \ldots, h_{k_d}(x_d, x_{d+1}))$ for $k_j \in K$ for $j = 1, \ldots, d$ and $v = (k_1, k_2, \ldots, k_d)$. Fig. 1 shows an example for $d = 4$.

It is obvious that there exists a probabilistic polynomial time algorithm M_G which, with inputs $v = (k_1, \ldots, k_d) \in K^d$ and $x = (x_1, \ldots, x_{d+1}) \in X^{d+1}$, outputs $g_v(x)$. M_G simply runs $\mathsf{M}_H(k_i, (x_i, x_{i+1}))$ for $i = 1, 2, \ldots, d$.

Let K_G be an algorithm which, with an input 1^n, runs $\mathsf{K}_H(1^n)$ d times independently. K_G is a probabilistic polynomial time algorithm for sampling an index of the hash functions in $G_{n,d}$ if d is a polynomial in n.

Lemma 1. *For every $g_v \in G_{n,d}$, if $g_v(x_1, x_2, \ldots, x_{d+1}) = g_v(x'_1, x'_2, \ldots, x'_{d+1})$ and $(x_1, x_2, \ldots, x_{d+1}) \neq (x'_1, x'_2, \ldots, x'_{d+1})$, then $x_j \neq x'_j$ for $j = 1, 2, \ldots, d+1$.*

Proof. This lemma is obvious from the assumption on H_n that both $h_k(x, \cdot)$ and $h_k(\cdot, x)$ are permutations for every k and x. □

In the remaining part, it is shown that $\{G_{n,d}\}_{n\in\mathbb{N}}$ is strongly CR with respect to the algorithm K_G for every weakly CR family $\{H_n\}_{n\in\mathbb{N}}$, where d is determined based on the weakness of CR of $\{H_n\}_{n\in\mathbb{N}}$.

Let $p(n)$ be the polynomial such that, for every probabilistic polynomial time algorithm F_H and every sufficiently large n,

$$\sum_{k\in K} \Pr[\mathsf{K}_H(1^n) = k] \; \Pr[\mathsf{F}_H(k) \text{ fails}] \geq \frac{1}{p(n)}.$$

Let $d = n\,p(n)$ and d is denoted by $d(n)$ to make it explicit that it depends on n.

Suppose that $\{G_{n,d(n)}\}_{n\in\mathbb{N}}$ is not strongly CR, that is, there exists a probabilistic polynomial time algorithm A_G and a polynomial q such that

$$\sum_{v\in K^{d(n)}} \Pr[\mathsf{K}_G(1^n) = v] \; \Pr[\mathsf{A}_G(v) \text{ succeeds}] \geq \frac{1}{q(n)} \tag{1}$$

for infinitely many n's.

Let C_H be a probabilistic polynomial time algorithm which, with an input $k \in K$, runs the following procedure B_H $b(n) = 2\,n\,d(n)\,q(n)$ times.

```
$\mathsf{B}_H(k)$ /* $k \in K$ */
{
  for ($j = 1$ to $d(n)$) {
    $(k_1, k_2, \ldots, k_{d(n)}) \leftarrow \mathsf{K}_G(1^n)$;
    $(x, x') \leftarrow \mathsf{A}_G(v)$; /* $v = (k_1, \ldots, k_{j-1}, k, k_{j+1}, \ldots, k_{d(n)})$ */
    if ($g_v(x) = g_v(x')$ and $x \neq x'$) {
      output $((x_j, x_{j+1}), (x'_j, x'_{j+1}))$;
      halt;
    }
  }
}
```

From Lemma 1, $\mathsf{B}_H(k)$ succeeds in finding a collision of h_k if $\mathsf{A}_G(v)$ succeeds in finding a collision of g_v. Let $B = \{k \mid k \in K,\ \Pr[\mathsf{B}_H(k) \text{ succeeds}] > n/b(n)\}$.

Lemma 2. *For every $k \in B$, $\Pr[\mathsf{C}_H(k)$ succeeds$] > 1 - \dfrac{1}{2^n}$.*

Proof. Since $\mathsf{C}_H(k)$ runs $\mathsf{B}_H(k)$ $b(n)$ times,

$$\Pr[\mathsf{C}_H(k) \text{ fails}] < \left(1 - \frac{n}{b(n)}\right)^{b(n)} < \frac{1}{2^n}.$$

□

Lemma 3. $\Pr[\mathsf{K}_H(1^n) \in B] > 1 - \dfrac{1}{2\,p(n)}$ *for infinitely many n's.*

Proof. Suppose that $\Pr[\mathsf{K}_H(1^n) \in B] \le 1 - \dfrac{1}{2\,p(n)}$ for every sufficiently large n. It is shown that there is a contradiction between this assumption and the inequality (1).

$$\begin{aligned}
&\sum_{v \in K^{d(n)}} \Pr[\mathsf{K}_G(1^n) = v] \Pr[\mathsf{A}_G(v) \text{ succeeds}] \\
&= \sum_{v \in K^{d(n)} - B^{d(n)}} \Pr[\mathsf{K}_G(1^n) = v] \Pr[\mathsf{A}_G(v) \text{ succeeds}] + \\
&\qquad \sum_{v \in B^{d(n)}} \Pr[\mathsf{K}_G(1^n) = v] \Pr[\mathsf{A}_G(v) \text{ succeeds}].
\end{aligned}$$

Let

$$\begin{aligned}
\sigma_1(n) &\overset{\text{def}}{=} \sum_{v \in K^{d(n)} - B^{d(n)}} \Pr[\mathsf{K}_G(1^n) = v] \Pr[\mathsf{A}_G(v) \text{ succeeds}], \\
\sigma_2(n) &\overset{\text{def}}{=} \sum_{v \in B^{d(n)}} \Pr[\mathsf{K}_G(1^n) = v] \Pr[\mathsf{A}_G(v) \text{ succeeds}].
\end{aligned}$$

Let $\mathsf{K}_H^{(l)}(1^n)$ represent the l-th run of $\mathsf{K}_H(1^n)$ of $\mathsf{K}_G(1^n)$ for $1 \le l \le d(n)$.

$$\begin{aligned}
\sigma_1(n) &= \sum_{v \in K^{d(n)} - B^{d(n)}} \left(\prod_{l=1}^{d(n)} \Pr[\mathsf{K}_H^{(l)}(1^n) = k_l] \right) \Pr[\mathsf{A}_G(v) \text{ succeeds}] \\
&\le \sum_{j=1}^{d(n)} \sum_{\substack{k_i \in K \\ 1 \le i \le d(n),\ i \ne j \\ k_j \in K - B}} \left(\prod_{l=1}^{d(n)} \Pr[\mathsf{K}_H^{(l)}(1^n) = k_l] \right) \Pr\left[\mathsf{A}_G(v) \text{ succeeds}\right] \\
&\overset{\text{def}}{=} \sum_{j=1}^{d(n)} \sigma_1'(n, j).
\end{aligned}$$

$$\begin{aligned}
&\sigma_1'(n, j) \\
&= \sum_{k_j \in K - B} \Pr[\mathsf{K}_H^{(j)}(1^n) = k_j] \sum_{\substack{k_i \in K \\ 1 \le i \le d(n), i \ne j}} \left(\prod_{\substack{l=1 \\ l \ne j}}^{d(n)} \Pr[\mathsf{K}_H^{(l)}(1^n) = k_l] \right) \Pr[\mathsf{A}_G(v) \text{ succeeds}] \\
&\le \max_{k_j \in K - B} \sum_{\substack{k_i \in K \\ 1 \le i \le d(n),\ i \ne j}} \left(\prod_{\substack{l=1 \\ l \ne j}}^{d(n)} \Pr[\mathsf{K}_H^{(l)}(1^n) = k_l] \right) \Pr[\mathsf{A}_G(v) \text{ succeeds}] \\
&\le \max_{k_j \in K - B} \Pr[\mathsf{B}_H(k_j) \text{ succeeds}] \\
&\le \frac{n}{b(n)}.
\end{aligned}$$

Thus, $\sigma_1(n) \leq \dfrac{n\,d(n)}{b(n)}$.

On the other hand, from the assumption that $\Pr[\mathsf{K}_H(1^n) \in B] \leq 1 - \dfrac{1}{2\,p(n)}$,

$$\sigma_2(n) \leq \sum_{v \in B^{d(n)}} \Pr[\mathsf{K}_G(1^n) = v] \leq \left(1 - \frac{1}{2p(n)}\right)^{d(n)} < \frac{1}{2^{n/2}} < \frac{n\,d(n)}{b(n)}$$

for every sufficiently large n.

Hence,

$$\sum_{v \in K^{d(n)}} \Pr[\mathsf{K}_G(1^n) = v] \Pr[\mathsf{A}_G(v) \text{ succeeds}] < \frac{2\,n\,d(n)}{b(n)} < \frac{1}{q(n)},$$

which causes a contradiction. □

From Lemmas 2 and 3, for infinitely many n's,

$$\begin{aligned}
&\sum_{k \in K} \Pr[\mathsf{K}_H(1^n) = k] \Pr[\mathsf{C}_H(k) \text{ succeeds}] \\
&\geq \sum_{k \in B} \Pr[\mathsf{K}_H(1^n) = k] \Pr[\mathsf{C}_H(k) \text{ succeeds}] \\
&> \left(1 - \frac{1}{2^n}\right) \sum_{k \in B} \Pr[\mathsf{K}_H(1^n) = k] \\
&> \left(1 - \frac{1}{2^n}\right)\left(1 - \frac{1}{2\,p(n)}\right) \\
&> 1 - \frac{1}{p(n)},
\end{aligned}$$

which contradicts the assumption that every probabilistic polynomial time algorithm fails to find a collision with probability at least $1/p(n)$ for every sufficiently large n. □

5 An Example of A CR Family of Hash Functions with The Property in Theorem 2

Hash functions based on discrete logarithms were proposed in [3]. A family of them is strongly CR if the discrete logarithm problem is intractable. It also satisfies the property in Theorem 2.

The CR family of hash functions based on discrete logarithms is defined in the followings. Precisely speaking, the definition of index-sampling algorithm is different from the one in Definitions 1 and 2.

For a positive integer a, let $\mathbb{Z}_a = \{0, 1, 2, \ldots, a - 1\}$ and $\mathbb{Z}_a^* = \{z \,|\, z \in \mathbb{Z}_a \wedge \gcd(z, a) = 1\}$.

S_H is a probabilistic polynomial time algorithm which, with an input 1^n, produces p and α, where p is an n-bit prime such that $q = (p-1)/2$ is a prime and α is an element of order q in $\mathbb{Z}_p^*$. p and α are shared by all the hash functions corresponding to the parameter n. K_H is a probabilistic polynomial time algorithm which, with inputs p and α, produces an element β of order q in $\mathbb{Z}_p^*$ at random. β is regarded as an index.

Let $G_p = \{\alpha^s \mid s \in \mathbb{Z}_q\}$. The family of hash functions $\{H_n^{(p,\alpha)}\}_{n\in\mathbb{N}}$ is defined by

$$H_n^{(p,\alpha)} = \{h_\beta \mid h_\beta : \mathbb{Z}_q \times \mathbb{Z}_q \to G_p,\ \beta \in G_p,\ h_\beta(x_1, x_2) = \alpha^{x_1}\beta^{x_2} \bmod p\}.$$

$\{H_n^{(p,\alpha)}\}_{n\in\mathbb{N}}$ is CR with respect to the algorithms S_H and K_H if the following discrete logarithm problem is intractable.

> For given p, α, β, compute $\log_\alpha \beta \bmod p$, where p and α is the output of $\mathsf{S}_H(1^n)$, β is the output of $\mathsf{K}_H(p, \alpha)$. The probability of success in computing the discrete logarithm is taken over the coin tosses of S_H, K_H and the algorithm computing the discrete logarithm.

It is easy to see that both $h_\beta(x, \cdot)$ and $h_\beta(\cdot, x)$ are permutations for every $x \in \mathbb{Z}_q$ and $\beta \in G_p$.

A weakly CR family of hash functions $\{F_n^{(p,\alpha)}\}_{n\in\mathbb{N}}$ can be constructed from $\{H_n^{(p,\alpha)}\}_{n\in\mathbb{N}}$ with the technique in Section 3. Let S_F be equivalent to S_H. Let K_F be a probabilistic polynomial time algorithm which, with inputs p and α, proceeds as follows.

1. It selects $u \in \{0,1\}^{\lceil \log n \rceil}$ at random.
2. If $u = 0^{\lceil \log n \rceil}$, it randomly selects $\beta \in G_p$ and outputs it. Otherwise, it outputs α.

$\{F_n^{(p,\alpha)}\}_{n\in\mathbb{N}}$ is not strongly CR because $h_\alpha(x_1, x_2) = \alpha^{x_1+x_2} \bmod p$ is selected with the probability more than $1 - 1/n$. It is easy to find a collision of h_α.

6 Conclusion

In this article, a new definition of weak CR of hash functions has been presented. Then, it has been shown that there really exists a weakly CR family of hash functions if there exists a CR family of hash functions. A method has also been presented to construct a strongly CR family of hash functions from a weakly CR one. Though this method is applicable only to a certain kind of families, a (weakly) CR family of hash functions has been presented to which it is applicable.

Weak second-preimage resistance can also be defined and the same kind of results are obtained in the similar way. These discussions are included in the final version.

Acknowledgements

The author thanks Masahiro Mambo for his valuable comments.

References

[1] J. H. An and M. Bellare. Constructing VIL-MACs from FIL-MACs: Message authentication under weakened assumptions. In *CRYPTO'99 Proceedings*, pages 252–269, 1999. Lecture Notes in Computer Science 1666. 87

[2] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO'96 Proceedings*, pages 1–15, 1996. Lecture Notes in Computer Science 1109. 87

[3] D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *CRYPTO'91 Proceedings*, pages 470–484, 1992. Lecture Notes in Computer Science 576. 88, 94

[4] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001. 88

[5] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. 87

[6] B. Preneel. The state of cryptographic hash functions. In *Lectures on Data Security*, pages 158–182, 1998. Lecture Notes in Computer Science 1561. 87

[7] D. R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *EUROCRYPT'98 Proceedings*, pages 334–345, 1998. Lecture Notes in Computer Science 1403. 88

Implementation of Tate Pairing on Hyperelliptic Curves of Genus 2

YoungJu Choie and Eunjeong Lee*

Dept. of Mathematics
POSTECH, Pohang, Korea
{yjc,ejlee}@postech.ac.kr

Abstract. Since Tate pairing was suggested to construct a cryptosystem, fast computation of Tate pairing has been researched recently. Barreto et. al[3] and Galbraith[8] provided efficient algorithms for Tate pairing on $y^2 = x^3 - x + b$ in characteristic 3 and Duursma and Lee[6] gave a closed formula for Tate pairing on $y^2 = x^p - x + d$ in characteristic p. In this paper, we present completely general and explicit formulae for computing of Tate pairing on hyperelliptic curves of genus 2. We have computed Tate parings on a supersingular hyperelliptic curve over prime fields and the detailed algorithms are explained. This is the first attempt to present the implementation results for Tate pairing on a hyperelliptic curve of genus bigger than 1.

Keywords. elliptic curve cryptosystem, Tate pairing implementation, hyperelliptic curve cryptosystem

1 Introduction

Since Weil pairing was proposed to cryptanalysis[16], Weil pairing and Tate pairing have contributed to two different aspects in cryptography community; one is attacking a cryptosystem[7] and the other side is building a cryptosystem[1]. Recently, the cryptosystem based on pairings becomes one of the most active research fields ([3],[2],[6],[9],[18],[21]). Tate pairing can be computed using an algorithm first suggested by Miller [14] which is described in [2], [3] and [9] for the case of elliptic curves. Miller algorithm on elliptic curves is basically the usual scalar point multiplication with an evaluation of certain intermediate rational functions which are straight lines used in the addition process.

While the hyperelliptic curve cryptosystem(HEC) is attractive because of the short key length, it seems an actual system is less efficient than that of elliptic curve[22]. However, [15], [13] and [19] gave the explicit formulae for group operation on the divisor class group of hyperelliptic curves of genus greater than 1. They found HEC can be more practical depending on platform and based library. However, the Miller algorithm on hyperelliptic curves is more complicated than the elliptic curve case because we must consider divisors instead of points.

* This work was partially supported by University ITRC fund.

This paper presents the explicit methods to implement Tate pairing on hyperelliptic curves of genus 2. We describe Miller algorithm for hyperelliptic curves in a detailed way and state some useful facts. Furthermore, the improved addition formulae on the divisor class group of hyperelliptic curves are given and an explicit method of deriving the rational functions $c, d \in \mathbb{F}_q[x, y]$ satisfying relation $\bar{D_3} + (c/d) = \bar{D_1} + \bar{D_2}$, has been suggested, which plays a key role in Miller algorithm. It turns out that the computing costs for the addition with the suggested new formulae are better than those given in [13] and [15].

The discrete logarithm problem in $J_H(\mathbb{F}_q)$ can be feasibly solved when the image of the pairing belongs to too small field $\mathbb{F}^*_{q^k}$, where k is called "security multiplier". Thus, it is necessary that $\#J_H(\mathbb{F}_q)$ should be divisible by a large prime($\approx 2^{160}$) which does not divide $q^k - 1$ for any small values k to obtain high security. On the contrary, for cryptographic applications such as identity based encryption schemes, Tate pairing needs to be mapped into not too large finite field $\mathbb{F}^*_{q^k}$ for the computational efficiency. Thus, it is of interest to produce families of hyperelliptic curves for which this "security multiplier" k is not too large, but not too small. To obtain a curve which satisfies an appropriate "security multiplier", it is natural to consider supersingular hyperelliptic curves[20]. According to [8], the security multiplier of supersingular hyperelliptic curves of genus 2 is bounded by 12 and if curve is defined over odd characteristic field then the maximal security multiplier is 6 [20]. Barreto et. al[3] and Galbraith[8] provided efficient algorithms for Tate pairing on $y^2 = x^3 - x + b$ in characteristic 3 with security mulitiplier 6 and Duursma and Lee[6] gave a closed formula for Tate pairing on $y^2 = x^p - x + d$ in characteristic p, $p \equiv 3 \pmod 4$, with security multiplier $2p$. However, since most common cryptosystems have developed based on binary fields or large prime fields, it is valuable to consider implementation of Tate pairing over such fields.

In this paper, we have implemented Tate pairing on hyperelliptic curves over large prime fields. Specifically, Tate pairing on the supersingular hyperelliptic curve $y^2 = x^5 + a$ over prime field $\mathbb{F}_p, p \equiv 2, 3 \pmod 5$, has been implemented. The security multiplier of the above curves is 4 which is the maximum among the known supersingular hyperelliptic curves over large prime fields.

Sections 2 and 3 describe the necessary definitions and properties about hyperelliptic curves and Tate pairing. Section 4 is about Miller algorithm. In Section 5 Tate paring on the supersingular hyperelliptic curves $H : y^2 = x^5 + a$ over large prime fields $\mathbb{F}_p, p \equiv 2, 3 \pmod 5$ has been computed. In the final section, concluding remark is given. Appendix A gives a proof of Lemma and Appendix B summarizes the addition law on the divisor class group and gives the explicit formulae for obtaining rational functions appearing in Miller algorithm.

2 Preliminaries

In this section, we recall the basic definitions and properties(see [12],[23] for further details).

Let $\mathbb{F}_q$ be a finite field and $\bar{\mathbb{F}}_q$ the algebraic closure of $\mathbb{F}_q$ with $q = p^n$, prime $p > 2$.

Definition 1 *Let $K/\mathbb{F}_q$ be a quadratic function field defined by $H/\mathbb{F}_q : y^2 + h(x)y = F(x)$, where $F(x) \in \mathbb{F}_q[x]$ is a monic polynomial with* $\deg(F) = 5$, *$h(x) \in \mathbb{F}_q[x], \deg(h) \leq 2$ and there are no singular points on H. The curve $H/\mathbb{F}_q$ associated to this function field is called a hyperelliptic curve of genus 2 defined over $\mathbb{F}_q$.*

Now consider $H = \{(a, b) \in \bar{\mathbb{F}}_q \times \bar{\mathbb{F}}_q | b^2 + h(a)b = F(a)\} \cup \{\mathcal{O}\}$ and let $H(\mathbb{F}_q) := H \cap (\mathbb{F}_q \times \mathbb{F}_q)$ be a set of rational points on H with infinite point $\mathcal{O}$.

For a free abelian group

$$\mathrm{Div}(K) = \{D | D = \sum_{P \in H} n_P P, n_P \in \mathbb{Z} \text{ and } n_P = 0 \text{ for almost all points } P \in H\},$$

consider a subgroup

$$\mathrm{Div}_0(K) = \{D = \sum_{P \in H} n_P P | \deg(D) = \sum_{P \in H} n_P = 0\}$$

which is called a group of zero divisors. An element $D \in \mathrm{Div}(K)$ is called a *divisor*. If $n_P \geq 0$, D is called *effective*. The *support* of D is defined as *supp*$(D) = \{P | D = \sum_{P \in H} n_P P \text{ such that } n_P \neq 0\}$ and the greatest common divisor of $D_1 = \sum_{P \in H} m_P P$ and $D_2 = \sum_{P \in H} n_P P$ in $\mathrm{Div}(K)$ is

$$\text{g.c.d.}(D_1, D_2) = \sum_{P \in H} min(m_P, n_P)P - (\sum_{P \in H} min(m_P, n_P))\mathcal{O}.$$

It is well-known that the set of principle divisors $\mathbb{P}_H = \{\mathrm{div}(g) | \mathrm{div}(g) = \sum_{P \in H} v_P(g)P, g \in K\}$, where v is a valuation map from K to $\mathbb{Z}$, forms a subgroup of $\mathrm{Div}_0(K)$. Two divisors D_1 and $D_2 \in \mathrm{Div}_0$ are said to be equivalent, $D_1 \sim D_2$, if $D_1 = D_2 + \mathrm{div}(f)$ for some $f \in K^*$. The set of equivalence classes forms a divisor class group $J_H = \mathrm{Div}_0(K)/\mathbb{P}_H$.

It is well known that each divisor class can be uniquely represented by the *reduced divisor* by means of Mumford representation [17];

Theorem 2 (Reduced divisor [17], [12]) *Let K be the function field given by $H : y^2 + h(x)y = F(x)$ where $h(x) = h_2x^2 + h_1x + h_0, F(x) = x^5 + \sum_{i=0}^{4} f_i x^i \in \mathbb{F}_q[x]$.*

1. *Then each divisor class of J_H can be represented by the reduced divisor*

$$D = \sum_{i=1}^{r} P_i - r\mathcal{O}, \text{ where } r = 1 \text{ or } 2, P_i \neq \mathcal{O}, P_i \in H.$$

2. *Put $P_i = (a_i, b_i)$, $1 \leq i \leq 2$. Let $u(x) = \prod_{i=1}^{r}(x - a_i), r = 1$ or 2. Then there exists a unique polynomial $v(x) \in \bar{\mathbb{F}}_q[x]$ satisfying* $\deg(v) < \deg(u) \leq 2, b_i = v(a_i)$ *and* $u(x) | (v(x)^2 + v(x)y - F(x))$. *Then* $D = \text{g.c.d.}(\mathrm{div}(u(x)), \mathrm{div}(v(x) - y))$.

A reduced divisor $\bar{D}$ will be denoted by $\bar{D} = [u_D, v_D] = D - \deg(D)\mathcal{O}$ with an effective divisor D.

3 Tate Pairing

In [7], Frey and Rück suggested that Tate paring can be used to attack a cryptosystem. On the other hand, recently Bonech and Franklin[1] used Weil pairing to construct an identity-based encryption and since then, many applications of Tate pairing have been developed (see [10]). In this section, we recall the definition of Tate pairing(see [7] for further details) and give useful remarks as the computational aspects.

Let m be a positive integer with $\gcd(m, q) = 1$ and k be the smallest integer such that $m|(q^k - 1)$ which is called *the security multiplier*. Let $J_H[m] = \{D \in J_H \mid mD = \mathcal{O}\}$. Tate pairing is a map

$$\begin{aligned} t : J_H[m] \times J_H(\mathbb{F}_{q^k})/mJ_H(\mathbb{F}_{q^k}) &\rightarrow \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^m \\ t(D, E) &= f_D(E') \end{aligned} \qquad (3.1)$$

where $\text{div}(f_D) = mD$ and $E' \sim E$ with $supp(E') \cap supp(\text{div}(f_D)) = \emptyset$.

It's well-known that Tate pairing satisfies the following three properties(see also [7]);

- (well-defined) For each $E \in J_H(\mathbb{F}_{q^k})$, $t(\mathcal{O}, E) \in (\mathbb{F}^*_{q^k})^m$, $\forall D \in J_H[m]$ and $\forall E \in mJ_H(\mathbb{F}_{q^k})$, $t(D, E) \in (\mathbb{F}^*_{q^k})^m$.
- (non-degeneracy) For each $D \in J_H[m] - \{\mathcal{O}\}$ there exits $E \in J_H(\mathbb{F}_{q^k})$ such that $t(D, E) \notin (\mathbb{F}^*_{q^k})^m$ (and vice versa).
- (bilinearity) For any integer n, $t(nD, E) = t(D, nE) = t(D, E)^n$ in $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^m$.

The following lemma suggests that one needs to take a divisor over the extension field of the defining field of the curve in order to get a nontrivial value of Tate-paring;

Lemma 3 *Let H be a hyperelliptic curve of genus 2 defined over $\mathbb{F}_q$ and m be a factor of $\#J_H(\mathbb{F}_q)$ with $\gcd(m, q) = \gcd(m, q-1) = 1$. For a rational function $f \in K^*$, take any divisor $E \in J_H(\mathbb{F}_q)$ such that $supp(E) \cap supp(\text{div}(f)) = \emptyset$. Then $f(E) \in (\mathbb{F}^*_{q^k})^m$.*

Proof. Since E and f are defined over $\mathbb{F}_q$, we have $f(E) \in \mathbb{F}^*_q$. From $\gcd(m, q-1) = 1$, $\mathbb{F}^*_q = (\mathbb{F}^*_q)^m \subset (\mathbb{F}^*_{q^k})^m$. □

To compute Tate pairing $t(\bar{D}, \bar{E})$ for reduced divisors $\bar{D}, \bar{E}$, one first needs to find a divisor E' such that $E' \sim \bar{E}$ with $supp(E') \cap supp(\text{div}(f_D)) = \emptyset$ where $\text{div}(f_D) = m\bar{D}$. So, in this case, one may take a random divisor S such that $E' = (\bar{E} + S) - S$ has disjoint support with $supp(\text{div}(f_D))$. Since $\mathcal{O} \in supp(\text{div}(f_D)) \cap supp(\bar{E})$, the translation seems a necessary process. However, Lemma 4 shows that, under certain condition, Tate paring can be computed by considering only effective divisor of $\bar{E}$(see also [2],[6]).

Lemma 4 *Let H be a hyperelliptic curve of genus 2 defined over $\mathbb{F}_q$ and m be a factor of $\#J_H(\mathbb{F}_q)$ with $\gcd(m,q) = \gcd(m,q-1) = 1$. Take a reduced divisor $\bar{D} = [u_D, v_D]$ such that $u_D, v_D \in \mathbb{F}_q[x]$ and $m\bar{D} = \mathrm{div}(f_D)$. Let $\bar{E} = E - 2\mathcal{O}$, where $E \in \mathrm{Div}(H(\mathbb{F}_{q^k}))$ is effective with $\deg(E) = 2$ and supp$(\mathrm{div}(f_D)) \cap$ supp$(E) = \emptyset$. Then, Tate paring $t(\bar{D}, \bar{E})$ can be computed as $t(\bar{D}, \bar{E}) = f_D(E) \in \mathbb{F}^*_{q^k}$.*

Proof. Take $x - a \in \mathbb{F}_q(x)$ such that a does not form a zero or pole of f_D and $\mathrm{div}(x-a) = R - 2\mathcal{O}$ is a $\mathbb{F}_q$−rational divisor. Note that $\bar{E} - \mathrm{div}(x - a) = E - R \sim E - 2\mathcal{O}$ and *supp*$(\bar{E} - \mathrm{div}(x-a)) \cap$ *supp*$(\mathrm{div}(f_D)) = \emptyset$. From Lemma 3, since $f_D(R) \in (\mathbb{F}^*_{q^k})^m$, we get $f_D(\bar{E}) = f_D(E-R) = f_D(E)/f_D(R) = f_D(E)$ in $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^m$. □

4 Miller Algorithm

Tate pairing can be computed using the algorithm first suggested by Miller [14]. Miller algorithm on elliptic curves is basically the usual scalar point multiplication with an evaluation of certain intermediate rational functions which are straight lines used in the addition process. The algorithm on hyperelliptic curves is more complicated than the elliptic curve case because we must consider divisors instead of points. In this section we give an explicit and general expression of the algorithm using divisors.

Algorithm 5 Computation of Tate pairing

```
Input: $\bar{D} = [u_D, v_D] \in J_H[m], \bar{E} = [u_E, v_E] \in J_H(\mathbb{F}_{q^k})$
Output: $f_D(E)$ where $\mathrm{div}(f_D) = m\bar{D}$
Step 1. Let $m = 2^s + \sum_{i=0}^{s-1} a_i 2^i$
Step 2. Set $f_c \leftarrow 1, f_d \leftarrow 1$, and $\bar{R} \leftarrow \bar{D}$
Step 3. For $i = 1$ to $s-1$ do
Step 3-1. compute $\bar{R}'$ and $c, d$ with $\bar{R}' = 2\bar{R} - \mathrm{div}(c/d)$
Step 3-2. $f_c \leftarrow f_c^2 \cdot c(E),\ \ f_d \leftarrow f_d^2 \cdot d(E),\ \bar{R} \leftarrow \bar{R}'$
Step 3-3. if $a_i = 1$, compute $\bar{R}'$ and $c, d$ with $\bar{R}' = \bar{R} + \bar{D} - \mathrm{div}(c/d)$.
Step 3-4. $f_c \leftarrow f_c \cdot c(E),\ f_d \leftarrow f_d \cdot d(E),\ \bar{R} \leftarrow \bar{R}'$
Step 4. Print out $f_c/f_d$
```

The main steps in Algorithm 5 are computing the rational functions $c, d \in \mathbb{F}_q(x, y)$ and evaluating the functions at a divisor E defined on the extension field $\mathbb{F}_{q^k}$.

4.1 Intermediate Rational Function

The following Algorithm 6 was proposed in [14] to find rational functions c and d such that $\bar{D}_3 + \mathrm{div}(c/d) = \bar{D}_1 + \bar{D}_2$ with given two reduced divisors $\bar{D}_1 = D_1 - \deg(D_1)\mathcal{O} = [u_1, v_1], \bar{D}_2 = D_2 - \deg(D_2)\mathcal{O} = [u_2, v_2] \in J_H(\mathbb{F}_q)$ and $D_1, D_2 \geq 0$.

For a divisor $D \in \mathrm{Div}(K)$, define a set $L(D) := \{f \in K \,|\, \mathrm{div}(f) \geq -D\} \cup \{0\}$.

Table 1. Comparison the cost of operations in J_H

	addition	doubling	$l(x)$
[15]	1I, 2S, 24M	1I, 4S, 23M	3M
[13]	1I, 3S, 22M	1I, 5S, 22M	3M
ours	1I, 2S, 23M	1I, 5S, 23M	no cost

Algorithm 6 [14]
Input: $\bar{D}_1 = [u_1, v_1], \bar{D}_2 = [u_2, v_2] \in J_H(\mathbb{F}_q)$
Output: $\bar{D}_3 = [u_3, v_3]$ *and* c, d *such that* $\bar{D}_3 + \mathrm{div}(c/d) = \bar{D}_1 + \bar{D}_2$
Step 1. *Find* $c(x,y) \in L(2\mathcal{O} - \bar{D}_1 - \bar{D}_2)$.
Step 2. *Compute the divisor such that* $D' = 2\mathcal{O} - \bar{D}_1 - \bar{D}_2 + \mathrm{div}(c)$.
Step 3. *Find* $d(x,y) \in L(4\mathcal{O} - D')$.
Step 4. *Compute the divisor* D_3 *such that* $D_3 = 4\mathcal{O} - D' + \mathrm{div}(d)$.

We suggest the following lemma which describes how to find c and d explicitly in Algorithm 6.

Lemma 7 *Let* $\bar{D}_1 = [u_1, v_1], \bar{D}_2 = [u_2, v_2]$ *be reduced divisors in* J_H *and* $u_2(x) = x^2 + u_{21}x + u_{20}$. *The function* c *and* d *in Algorithm 6 can be found as follows;*

1. *If* $\gcd(u_1, u_2, v_1 + v_2 + h) = 1$, *then* $c(x,y) = y - l(x)$ *where* $l(x)$ *satisfies*

$$\begin{aligned} v_1 &\equiv l \pmod{u_1} \\ v_2 &\equiv l \pmod{u_2} \\ F &\equiv l^2 + h \cdot l \pmod{u_1 u_2}. \end{aligned} \tag{4.1}$$

   Furthermore, $d(x,y) = monic(\frac{F - l^2 - hl}{u_1 u_2}) = u_3$.
2. *If* $\deg(u_1) = 1$ *and* $\gcd(u_1, u_2, v_1 + v_2 + h) \neq 1$, *then* $c(x,y) = (x - a_1)(x - a_4)$ *and* $d(x,y) = x - a_4$. *Here,* $x - a_1 = \gcd(u_1, u_2)$ *and* $a_4 = -u_{21} - a_1$.
3. *Let* $u_1(x) = x^2 + u_{11}x + u_{10}$. *If* $\gcd(u_1, u_2, v_1 + v_2 + h) = x - a_1$, *then* $c(x,y) = (x - a_1)(x - a_2)(x - a_4)$ *and* $d(x,y) = (x - a_2)(x - a_4)$. *Here,* $a_2 = -u_{11} - a_1$ *and* $a_4 = -u_{21} - a_1$.
4. *If* $\deg(u_1) = 2$ *and* $\gcd(u_1, u_2, v_1 + v_2 + h) = u_1$, *then* $c(x,y) = u_1$ *and* $d(x,y) = 1$..

Proof. See Appendix A.

Remark 8 1. *In Lemma 7, let* $\deg(u_1) = \deg(u_2) = 1$ *and* $\bar{D}_1 = [u_1, v_1], \bar{D}_2 = [u_2, v_2]$. *If* $D_2 = -D_1$, *then* $c(x,y) = x - x_1$ *and* $d(x,y) = 1$. *Otherwise,* $c(x,y) = 1$ *and* $d(x,y) = u_1 u_2$.
2. *In Appendix B, the formulae for* l, u_3 *and* v_3 *in Lemma7 are explicitly given for generic case(for cases of reduced divisors, see [13]). Table 1 describes the computing cost about addition and doubling.*
   Note that, [15] and [13] gives the formulae for $\bar{D}_3$. *However, one may need to do more work to get* $c(x,y) = y - l(x)$ *of Algorithm 6. Here, the formulae in Appendix B give the direct formulae for* l *and* $\bar{D}_3$ *with less cost than [15] and [13].*

4.2 Evaluation

The evaluation of a rational function f at a divisor $E, E = [u_E, v_E] = \sum_{P \in H, n_P \in \mathbb{Z}} n_P P \in \mathrm{Div}(K)$, is meant by

$$f(E) = \prod_{P \in H} f(P)^{n_P}. \tag{4.2}$$

So, to compute $t(\bar{D}, \bar{E})$ in Algorithm 5, one needs to know the roots of u_E, and those are generally in the extension field. However, the following corollary suggests that $f(E)$ can be evaluated without computing roots;

Corollary 9 *Let $f \in K$ be a rational function and $\bar{E} = [u_E, v_E]$ be a reduced divisor such that $supp(\mathrm{div}(f)) \cap supp(E) = \emptyset$. Then $f(E)$ can be represented as a function of the coefficients of u_E and v_E.*

Proof. See the Table 2 and Table 3 in Section 5.2.

5 Implementation of Tate Pairing over Large Prime Field

In this section we describe an implementation result of Tate pairing on a hyperelliptic curve of genus 2 using the above suggested algorithms.

5.1 Choice of Curves

It is of interest to produce families of hyperelliptic curves for which this "security multiplier" k is not too large, but not too small. To obtain a curve which satisfies an appropriate "security multiplier", it is natural to consider supersingular hyperelliptic curves[20]. According to [8], the security multiplier of supersingular hyperelliptic curves of genus 2 is bounded by 12 and if curve is defined over odd characteristic field then the maximal security multiplier is 6 [20]. The security multiplier 4 is the maximum among the known supersingular hyperelliptic curves of genus 2 defined over large prime fields. Since most cryptosystems have developed based on binary fields or large prime fields, it is valuable to consider implementation over such fields.

Remark 10 *Lemma 3 suggests that we need to find a method to choose a reduced divisor which belongs to the larger group $J_H(\mathbb{F}_{q^k})$ to get a nontrivial value of Tate pairing. In general, there is no known deterministic method to find divisors D, E to get a nontrivial value of $t(\bar{D}, \bar{E})$, that is, $t(D, E) \notin (\mathbb{F}_{q^k}^*)^m$. However, supersingular curves provide a valuable technique to get the nontrivial value for $t(\bar{D}, \bar{E})$ ([24]).*

In this paper, we have implemented Tate pairing on

$$H : y^2 = x^5 + a,\ a \in \mathbb{F}_p^*,\ \ p \equiv 2, 3 \pmod 5. \tag{5.1}$$

This curve is supersingular with $k = 4$(This curve was used in [5]).

Note that H has the following endomorphism, called a "*distortion map*";

$$\phi : H \to H \text{ defined by } \phi((x, y)) = (\zeta x, y),$$

where ζ is the primitive 5th root of unity.

Let m be a prime factor of p^2+1 with $\gcd(m, p) = \gcd(p-1) = 1$ and $\tilde{\phi}$ be the induced map of ϕ on $\mathrm{Div}(K)$. Define the twisted Tate pairing as (see [6],[8],[10]),

$$\begin{aligned} \hat{t} : J_H[m] \times J_H(\mathbb{F}_{p^4})/mJ_H(\mathbb{F}_{p^4}) &\to \mathbb{F}_{p^4}^*/(\mathbb{F}_{p^4}^*)^m \\ \hat{t}(\bar{D}, \bar{E}) &:= t(\bar{D}, \tilde{\phi}(\bar{E})). \end{aligned}$$

For a reduced divisor $\bar{D} = D - 2\mathcal{O} = [u_D, v_D] \in J_H(\mathbb{F}_q)[m]$ where $u_D(0) \neq 0$, note that $\tilde{\phi}(\bar{D}) \in J_H(\mathbb{F}_{p^4}) - J_H(\mathbb{F}_q)$. Hence, $\mathit{supp}(\mathrm{div}(f)) \cap \mathit{supp}(\tilde{\phi}(D)) = \emptyset$ for $\mathrm{div}(f_D) = m\bar{D}$ and $\hat{t}(\bar{D}, \bar{D}) = f_D(\tilde{\phi}(D))$, where $\mathrm{div}(f) = m\bar{D}$.

Remark 11 *As emphasized in [8], Tate pairing is defined up to a multiple by an mth power in $\mathbb{F}_{p^4}$. For a unique value, it is necessary to exponentiate the value of Tate pairing to the power $(p^k - 1)/m$ in $\mathbb{F}_{p^4}$. Using Karatsuba's idea[11], the multiplication in $\mathbb{F}_{p^4}$ needs 9 multiplications and the squaring needs 8 multiplications over $\mathbb{F}_p$.*

5.2 Computation of Tate Pairing

Computing Tate pairing is mainly an evaluation of $t(\bar{D}, \tilde{\phi}(\bar{E})) = f_D(\tilde{\phi}(E))$ for $f_D \in K$ such that $\mathrm{div}(f_D) = m\bar{D}$ using Algorithm 5. The value is obtained by successive evaluation of c and d at $\tilde{\phi}(E)$, which is defined on the extension field $\mathbb{F}_{p^4}$.

To investigate the cost for computing $t(\bar{D}, \tilde{\phi}(\bar{E}))$ using Algorithm 5, let

- T_D : time for doubling and intermediate rational function
- T_A : time for addition and intermediate rational function
- T_c and T_d : time for evaluation of c, d at $\tilde{\phi}(E)$
- T_{sk} and T_{mk}: time for squaring and multiplication in $\mathbb{F}_{q^k}$.

Then the total cost is given by

$$\log(m) \cdot (T_D + T_c + T_d + 2T_{sk} + 2T_{mk}) + 1/2 \log(m) \cdot (T_A + T_c + T_d + 2T_{mk}).$$

Through the previous sections, we have discussed

$$\begin{aligned} &T_D = 1I + 23M + 5S, T_A = 1I + 23M + 2S \quad \text{for most common case} \\ &T_{sk} = 8M,\ \ T_{mk} = 9M \end{aligned}$$

Table 2. Evaluation $c(\tilde{\phi}(E)$

Input: $c(x) = y - l(x), d(x) = x^2 + d_1x + d_0$ and $\bar{E} = [u_E, v_E]$ where $l(x) = sx^3 + l_2x^2 + l_1x + l_0$
Output: $f_c = c(\tilde{\phi}(E)) = f_{c3}\zeta^3 + f_{c2}\zeta^2 + f_{c1}\zeta + f_{c0}$ and $f_d = d(\tilde{\phi}(E)) = f_{d3}\zeta^3 + f_{d2}\zeta^2 + f_{d1}\zeta + f_{d0}$
$t_1 = s(2u_{E0} - u_{E1}^2), t_2 = u_{E0}l_2, t_3 = u_{E1}(l_0 - v_{E0}), t_4 = t_3 + u_{E0}(v_{E1} + l_1), t_5 = l_1u_{E0}$ $z_1 = (l_0 - v_{E0})^2, z_2 = su_{E0}, z_3 = l_2u_{E1}$ $f_{c3} = t_1t_4 + z_2t_3 - t_2(t_2 + l_1u_{E1}),\ \ f_{c2} = t_5(t_1 + l_1 - z_3) + z_3t_4 + z_1 - (t_2 + l_0 - v_{E0})^2$ $z_3 = t_5t_1$ $f_{c1} = z_3 - l_1(2(t_4 - t_5) - t_3) + z_2^2u_{E0} - t_2^2,\ \ f_{c0} = z_3 + v_{E1}(t_4 - t_5) + z_1 - t_2(t_2 + z_2u_{E1})$
Cost : 19M, 5S
$f_{d3} = -u_{E0}(d_1u_{E1} + u_{E0}),\ \ f_{d2} = d_0u_{E1}^2 + u_{E0} \cdot (d_1^2 - 2d_0 - u_{E0})$ $f_{d1} = f_{d3} - (d_1u_{E1}) \cdot (d_0 - u_{E0}),\ \ f_{d0} = (d_0 + u_{E0}) \cdot (d_0 - u_{E0})$
Cost : 6M, 2S

From the definition (4.2), the evaluation $c(\tilde{\phi}(E))$ and $d(\tilde{\phi}(E))$ requires the roots of $u(x) = 0$. On the contrary to the case of elliptic curves, the roots generally belong to the extension field $\mathbb{F}_{p^2}$ for the case of hyperelliptic curves of genus 2. With assumption that the roots are contained in $\mathbb{F}_p$, the cost of computing $c(\tilde{\phi}(E))$ and $d(\tilde{\phi}(E))$ for general c and d, i.e., $c(x, y) = y - sx^3 - l_2x^2 - l_1x - l_0$ and $d(x) = x^2 + d_1x + d_0$ is 25M and 4S.

Therefore, the computation of $f_D(E)$ by Algorithm 5 takes about

$$2\log(p)M + \log(m) \cdot (1I + 82M + 9S) + 1/2\log(m) \cdot (1I + 66M + 6S). \quad (5.2)$$

where $2\log(p)M$ is for Legendre symbol and square root in $\mathbb{F}_p$.

Generally, computing Tate pairing needs the field operations in the extension field, which are, of course, more time consuming. Thus, the suggested explicit formulae, which use only coefficients of u_E and v_E, given in the following tables are useful tool for speeding up the algorithms.

Using a Divisor Representation Table 2 describes an explicit formula for $c(\tilde{\phi}(E))$ where $\bar{E} = [u_E, v_E]$ and $c(x, y) = y - l(x)$ of $\deg(l) = 3$. Note that this type of $c(x, y)$ occurs most commonly in the applications. Computing cost for Algorithm 5 using Table 2 is about

$$\log(m) \cdot (1I + 82M + 12S) + 1/2\log(m) \cdot (1I + 66M + 9S).$$

Using a Divisor with Precomputation An evaluation can be more efficient, if we permit some precomputation and spaces. By reorganizing the formula in Table 2, the number of operations in Algorithm 5 can be further reduced. The formula is given in Table 3.

The precomputation takes 8M and 3S and each evaluation takes 22M and 5S.

Thus, the total cost is

$$8M + 3S + \log(m) \cdot (1I + 79M + 10S) + 1/2\log(m) \cdot (1I + 63M + 7S). \quad (5.3)$$

Table 3. Evaluation $c(\tilde{\phi}(E)$ with precomputation

Input: $c(x) = y - l(x)$ and $d(x) = x^2 + d_1x + d_0$ $\bar{E} = [u_E, v_E]$ where $l(x) = sx^3 + l_2x^2 + l_1x + l_0$ $t_1 = u_{E0}^2, t_2 = u_{E1}^2 - 2u_{E0}, t_3 = u_{E0}t_2, t_4 = u_{E1} \cdot (u_{E0} - t_2)$ $t_5 = u_{E_0}u_{E1}, t_6 = t_3v_{E1}, t_7 = 2u_{E0}v_{E1}, t_8 = v_{E1}u_{E1}, t_9 = v_{E1}^2u_{E0} - v_{E0}t_8$
Output: $f_c = c(\tilde{\phi}(E)) = f_{c3}\zeta^3 + f_{c2}\zeta^2 + f_{c1}\zeta + f_{c0}$ and $f_d = d(\tilde{\phi}(E)) = f_{d3}\zeta^3 + f_{d2}\zeta^2 + f_{d1}\zeta + f_{d0}$
$w_1 = l_2u_{E0}, w_2 = w_1u_{E1}, w_3 = w_1^2, w_4 = st_3, w_5 = su_{E0}$ $f_{c3} = l_1 \cdot (w_2 - w_4) + s \cdot (t_6 + t_4 \cdot (l_0 - v_{E0})) - w_3$ $f_{c2} = l_2 \cdot (t_5 + t_2 \cdot (l_0 - v_{E0})) + l_1 \cdot (l_1u_{E0} - w_4) - w_3$ $f_{c1} = l_1 \cdot (-t_7 - u_{E1} \cdot (l_0 - v_{E0}) - w_4) - w_3 + w_5^2u_{E0}$ $f_{c0} = t_9 + t_8l_0 - w_5 \cdot (w_2 + l_1t_2) - w_3 + (l_0 - v_{E0})^2$
Cost : 17M, 3S
$f_{d3} = -u_1t_5 - t_1,\ f_{d2} = u_0t_2 + (u_1^2 - 1) \cdot t_1,\ f_{d1} = -u_1u_0u_{E1} - t_1,\ f_{d0} = u_0^2 - t_1$
Cost : 5M, 2S

Table 4. The timing result

method	theoretical cost	timing result
with points	240I, 18912 M, 1920 S	594 ms
with divisors	240I, 18400 M, 2640 S	546 ms
with precomputation	240I, 17688 M, 2163 S	515 ms

5.3 The Implementation Results

Finally, we present the implementation result of Tate pairing using the detailed algorithms suggested through this paper for the following curve;

$$H/\mathbb{F}_p : y^2 = x^5 + a, p \equiv 2, 3 \pmod 5), p > 2.$$

Since this curve has 4 as security multiplier, the number of bits of prime p is chosen as 256 for high security. We take m as a prime of 160 bits such that $m | \#J_H(\mathbb{F}_p) = p^2 + 1$.

The elapsed time of field multiplication and inversion in a prime field $\mathbb{F}_p$ of MP-library[1] is as follows;

# of bits of p	Multiplication(M)	Inversion(I)	MI-ratio
256	6.3 μ	656 μ	104

The timings were performed on a 2 GHz Pentium IV with 256 Mb RAM and the language used was C. The compiler was Microsoft Visual C++ 6.0 with no speed optimizations on. No optimization algorithm for the field operation and scalar multiplication is adopted to give the general tips for the implementation of Tate pairing on hyperelliptic curves.

Table 4 presents the result of theoretical analysis and implemented results for $p \approx 2^{256}, p^k \sim 2^{1024}$ and $m \approx 2^{160}$. For a test, we have chosen a divisor D of which support is contained in $\mathbb{F}_p$. The timing result includes the exponentiation in $\mathbb{F}_{p^4}$.

[1] The MP-library is developed privately.

6 Conclusion and Future Work

In this paper, we suggest detailed methods to implement Tate pairing on hyperelliptic curves of genus 2. Specifically, Tate paring on the supersingular hyperelliptic curve $H : y^2 = x^5 + a$ over prime field $\mathbb{F}_p, p \equiv 2, 3 \pmod 5$ has been implemented.

For practical applications, it will be interesting to compare the speed of Tate pairings between hyperelliptic curves and elliptic curves. To do this, we need to optimize the algorithms of field operations and scalar multiplication on the divisor class group J_H. Finally, we report that we are in the progress for computing Tate paring of hyperelliptic curves over the even characteristic field.

References

[1] D. Boneh and M. Franklin, Identity-based Encryption from the Weil paring, *Advances in Cryptology-CRYPTO 2001*, LNCS **2139**, Springer-Verlag, (2001), pp.21-229, Springer-Verlag. 97, 100

[2] P. S.Barreto, H. Y.Kim, B. Lynn and M. Scott, Efficient Algorithms for Pairing-Based Cryptosystems, Cryptology eprint Archives, Available at http://eprint.iacr.org, (2002), Number 2002/008. 97, 100

[3] P. Barreto, B, Lynn and M, Scott, On the Selection of Pairing-Friendly Groups, Cryptology eprint Archives, Available at http://eprint.iacr.org, (2003), Number 2003/086. 97, 98

[4] D. G.Cantor, Computing in the Jacobian of a Hyperelliptic Curves, *Math. Comp*, **48**, No.177 (1987), pp.95-101.

[5] Y. Choie, E.Jeong and E. Lee, Supersingular Hyperelliptic Curves of Genus 2 over Finite Fields, Cryptology eprint Archives, Available at http://eprint.iacr.org, (2002), Number 2002/032. 104

[6] I. Duursma and H. Lee, Tate-priring implementations for tripartite key agreement, Cryptology eprint Archives, Available at http://eprint.iacr.org, (2003), Number 2003/053. 97, 98, 100, 104

[7] G. Frey and H-G. Rück, A remark concerning m-divisibility in the divisor class group of curves, *Math. Comp.* **62**, No.206 (1994), pp.865-874. 97, 100

[8] S. Galbraith, Supersingular curves in Cryptography, *Advances in Cryptology-AsiaCrypt'2001*, LNCS **2248**, Springer-Verlag, (2002), pp.495-513. 97, 98, 103, 104

[9] S. Galbraith, K. Harrison, and D. Soldera, Implementing the Tate pairing, ANTS-V, LNCS **2369**, Springer-Verlag, (2002), pp.324-337. 97

[10] A. Joux, A one round protocol for tripartite Diffie-Hellman, ANTS-IV, LNCS **1838**, Springer-Verlag, (2000), pp.385-393. 100, 104

[11] A. Karatsuba and Y. Ofman, Multiplication of Multidigit Numbers on Automata, *Sov. Phys.-Dokl. (Engl. transl.)*, **7**, No. 7 (1963), pp.595-596. 104

[12] N. Koblitz, *Algebraic aspects of cryptography*, Springer-Verlag (1998). 98, 99

[13] T. Lange, Efficient Arithmetic on Genus 2 Hyperelliptic Curves over Finite Fields via Explicit Formulae, Cryptology eprint Archives, Available at http://eprint.iacr.org, (2002), Number 2002/121. 97, 98, 102, 109

[14] V. Miller, Short Programs for Functions on Curves, Unpublished manuscript, 1986. 97, 101, 102

[15] Y. Miyamoto, H. Doi, K. Matsuo, J. Chao, and S. Tsuji, A fast addition algorithm of genus two hyperellipitc curve, In *Proc. of SCIS2002,* IEICE Japan, (2002), pp.497-502(in Japanese). 97, 98, 102

[16] A. J. Menezes, T. Okamoto and S.AQ. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, *IEEE Thans. Inf. Theory*, **39**, No.5 (1993), pp.1639-1646. 97

[17] D. Mumford, *Tata Lectures on Theta II*, Birkhäuser, 1984. 99

[18] K. G. Paterson, ID-based signature from pairings on elliptic curves, *Electronis Letters*, **38** No.18 (2002), pp.1025-1026. 97

[19] J. Pelzl, T. Wollinger, J. Guajardo and C. Paar, Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves (Update), Cryptology eprint Archives, Available at http://eprint.iacr.org, (2003), Number 2003/026. 97

[20] K. Rubin and A. Silverberg, The Best and Worst of Supersingular Abelian Varieties in Cryptology, Cryptology eprint Archives, Available at http://eprint.iacr.org, (2002), Number 2002/121. 98, 103

[21] R. Sakai and M. Kasahara, ID based Cryptosystems with Pairing on Elliptic Curve, Cryptology eprint Archives, Available at http://eprint.iacr.org, (2003), Number 2003/054. 97

[22] N. P.Smart, On the Performance of Hyperelliptic Cryptosystems, *Advances in Cryptology-Eurocrypt'99*, LNCS **1592**, Springer-Verlag, (1999), pp.165-175. 97

[23] H. Stichtenoth, *Algebraic Function Fields and Codes*, Springer Verlag (1993). 98

[24] E. R. Verheul, Evidence that XTR Is More Secure than Supersingular Elliptic Curve Cryptosystems,*Advances in Cryptology-AsiaCrypt'2001*, LNCS **2248**, Springer-Verlag, (2002), pp.433-551. 103

Appendix A: Proof of Lemma 7

Let $D' = 2\mathcal{O} - \bar{D}_1 - \bar{D}_2 + \mathrm{div}(c(x, y))$. Note that D' is an effective divisor and this means that c passes the points which support D_1, D_2.

(1) The case when $\deg(u_1) = 2$; Let $D_1 = P_1 + P_2 = (a_1, b_1) + (a_2, b_2), D_2 = P_3 + P_4 = (a_3, b_3) + (a_4, b_4)$. In this case $L(2\mathcal{O} - \bar{D}_1 - \bar{D}_2)$ is a subset of $L(6\mathcal{O})$ which is generated by $\{1, x, x^2, x^3, y\}$. Thus we can write $c(x, y) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4y$. Suppose $c_4 = 0$. Then, from $2\mathcal{O} - \bar{D}_1 - \bar{D}_2 = 6\mathcal{O} - P_1 - P_2 - P_3 - P_4 + \mathrm{div}(c) \geq 0$, P_1, P_2, P_3, P_4 are zeros of c. But, $c(x, y) = c_0x + c_1x + c_2x^2 + c_3x^3$ has only three zeros. This means P_3 or P_4 should equal to $-P_1$ or $-P_2$, say that$P_3 = -P_1$ or $P_4 = -P_2$. This is equivalent to $\gcd(u_1, u_2, v_1 + v_2 + h) \neq 1$. If $\gcd(u_1, u_2, v_1 + v_2 + h) = x - a_1$ then $P_3 = -P_1$ and $P_4 \neq -P_2$. This case yields $c(x, y) = (x - a_1)(x - a_2)(x - a_4)$ and $D' = -P_2 - P_4 + 4\mathcal{O}$. If $\gcd(u_1, u_2, v_1 + v_2 + h) = (x - a_1)(x - a_2)$ then $P_3 = -P_1$ and $P_4 = -P_2$ and this implies $c(x, y) = (x - a_1)(x - a_2)$ and $D' = 2\mathcal{O}$. As a result, if $\gcd(u_1, u_2, v_1 + v_2 + h) = 1$ then we can write $c(x, y) = c_0 + c_1x + c_2x^2 + c_3x^3 + y = y - l(x)$. Since P_i passes c, $c(a_i, b_i) = b_i - l(a_i) = 0$ and thus l satisfies the three congruence equations (4.1) by Lemma 3.3 in [p.162, 12] and has of the form $l(x) = (s_1x + s_0)u_2 + v_2$ for some $s_1, s_0 \in \mathbb{F}_q$.

The case when $\deg(u_1) = 1$; One can derive the conclusion by the similar way to the case (i). So, we omit the detailed proof.

(2) Formula for $d(x)$; The points supporting $D' = 2\mathcal{O} - \bar{D}_1 - \bar{D}_2 + \mathrm{div}(c)$ are zeros of c. Since c is a curve through supporting points of D_1, D_2, c meets the given hyperelliptic curve H and thus the supports of D' are other intersection points of c and the curve H. Since $D_3 + D' - 4\mathcal{O} = \mathrm{div}(d)$, D_3 and D' are opposite. This completes Lemma 7.

Appendix B: Formulae

The following tables describe explicit formulae for D_3 and $l(x)$ such that $\bar{D}_3 + \mathrm{div}(\frac{y-l(x)}{u_3}) = \bar{D}_1 + \bar{D}_2$ for the most common case (for cases see [13]). Here, $\bar{D}_i = [u_i, v_i], i = 1, 2, 3$ are reduced divisors in J_H for $H : y^2 + h(x)y = F(x)$ over $\mathbb{F}_q$ of genus 2. We take $\bar{D}_1 \neq \bar{D}_2$ in Table B.1 and the duplication is described in Table B.2. The number of field multiplications(M), squaring(S) and inversion(I) in $\mathbb{F}_q$ are listed.

Table 5. Addition formula when $\deg u_1 = \deg u_2 = 2, \gcd(u_1, u_2) = 1$

Input Output	$D_1 = [u_1, v_1], D_2 = [u_2, v_2]$ $u_1 = x^2 + u_{11}x + u_{10}, u_2 = x^2 + u_{21}x + u_{20}, v_1 = v_{11}x + v_{10}, v_2 = v_{21}x + v_{20}$ $\bar{D}_3 = [u_3, v_3]$
Step 1	Compute $r = res(u_1, u_2)$ $z_1 = u_{11} - u_{21}, z_2 = u_{20} - u_{10}, z_3 = u_{11}z_1$ $r = z_2(z_3 + z_2) + z_1^2 u_{10}$
Step 2	Compute almost inverse of $u_2 \pmod{u_1}$ $inv_1 = z_1, inv_0 = z_3 + z_2$
Step 3	Compute $s' = rs \equiv (v_1 - v_2)inv \pmod{u_1}$ $w_0 = v_{10} - v_{20}, w_1 = v_{11} - v_{21}, w_2 = inv_0 w_0, w_3 = inv_1 w_1$ $s'_1 = z_1 w_0 + z_2 w_1, s'_0 = w_2 - u_{10} w_3$ If $s_1 = 0$ then goto step 4'.
Step 4	Compute $s = s_1 x + s_0$ and s_1^{-1} $w_1 = (rs'_1)^{-1}(= 1/r^2 s_1), w_2 = s'_1 w_1 (= 1/r),$ $w_3 = r^2 w_1 (= 1/s_1), s_1 = s'_1 w_2, s_0 = s'_0 w_2$
Step 5	Compute $l(x) = su_2 + v_2 = s_1 x^3 + l_2 x^2 + l_1 x + l_0$ $l_2 = s_1 u_{21} + s_0, l_1 = (s_1 + s_0)(u_{21} + u_{20}) - s_1 u_{21} - s_0 u_{20} + v_{21}, l_0 = s_0 u_{20} + v_{20}$
Step 6	Compute $u_3 = monic(\frac{F - l^2 - h \cdot l}{u_1 u_2}) = x^2 + u_{31}x + u_{30}$ $s''_0 = w_3 s_0, w_1 = u_{11} + u_{21},\ \ u_{31} = s''_0 - z_1 - w_3^2 + h_2 w_3$ $u_{30} = s''_0 \cdot (s''_0 - 2u_{11}) + z_3 - u_{10} - u_{20} + w_3 \cdot (2l_1 + h_1 - h_2 w_1 - w_3 \cdot (f_4 - w_1 - h_2 l_2))$
Step 7	Compute $v_3 = -l - h \pmod{u_3}$ $w_1 = u_{31} s_1, w_2 = l_2 + h_2 - w_1, w_3 = u_{30} w_2$ $v_{31} = (u_{31} + u_{30})(w_2 + s_1) - w_3 - w_1 - l_1 - h_1, v_{30} = w_3 - l_0 - h_0$
Cost	23M, 2S, 1I
Step 4'	Compute $l(x) = s_0 u_2 + v_2$ $inv = 1/r, s_0 = s'_0 inv, l_1 = s_0 u_{21} + v_{21}, l_0 = s_0 u_{20} + v_{20}$
Step 5'	Compute $u_3 = monic(\frac{F - l^2 - h \cdot l}{u_1 u_2}) = x + u_{30}$ $u_{30} = f_4 - u_{21} - u_{11} - s_0^2 - h_2 s_0$
Step 6'	Compute $v_3 = -l - h \pmod{u_3} = v_{30}$ $v_{30} = u_{30}(l_1 + h_1 - u_{30}(s_0 + h_2)) - l_0 - h_0$
Cost	13M, 2S, 1I

Table 6. Doubling formula when $\deg u_1 = 2, \gcd(u_1, 2v_1 + h) = 1$

Input	$\bar{D_1} = [u_1, v_1]$ where $u_1 = x^2 + u_{11}x + u_{10}, v_1 = v_{11}x + v_{10}$
	$h = h_2x^2 + h_1x + h_0, f = x^5 + \sum_{i=0}^{4} f_i x^i$
Output	$\bar{D_3} = [u_3, v_3], l(x)$ such that $\bar{D_3} + \mathrm{div}((y - l)/u_3) = 2\bar{D_1}$

Step	Expression
1	Compute $\tilde{v_1} \equiv (h + 2v_1) \pmod{u_1} = \tilde{v}_{11}x + \tilde{v}_{10}$
	$\tilde{v}_{11} = h_1 + 2v_1 - h_2u_{11}, \tilde{v}_{10} = h_0 + 2v_{10} - h_2u_{10}$
2	Compute $r = res(u_1, \tilde{v_1})$
	$w_0 = v_{11}^2, w_1 = u_{11}^2, w_2 = \tilde{v}_{11}{}^2, w_3 = u_{11}\tilde{v}_{11}$
	$r = u_{10}w_2 + \tilde{v}_{10}(\tilde{v}_{10} - w_3)$
3	Compute almost inverse of $inv' = r \cdot (2v_1 + h)^{-1} \pmod{u_1}$
	$inv'_1 = -\tilde{v}_{11}, inv'_0 = \tilde{v}_{10} - w_3$
4	Compute $k' = \frac{F - v_1^2 - hv_1}{u_1} \pmod{u_1} = k'_1x + k'_0$
	$w_3 = f_3 + w_1, w_4 = 2u_{10}$
	$k'_1 = 2(w_1 - f_4u_{11}) + w_3 - w_4 - v_{11}h_2$
	$k'_0 = u_{11}(2w_4 - w_3 + f_4u_{11} + h_2v_{11}) + f_2 - w_0 - 2f_4u_{10} - v_{11}h_1 - v_{10}h_2$
5	Compute $s' = k' \cdot inv' \pmod{u_1}$
	$w_0 = k'_0 inv'_0, w_1 = k'_1 inv'_1$
	$s'_1 = \tilde{v}_{10}k'_1 - \tilde{v}_{11}k'_0, s'_0 = w_0 - u_{10}w_1$
	If $s'_1 = 0$ then goto step 6'.
6	Compute $s = s_1x + s_0$ and s_1^{-1}
	$w_1 = (rs'_1)^{-1}(= 1/r^2s_1), w_2 = s'_1w_1(= 1/r), w_3 = r^2w_1(= 1/s_1)$
	$s_1 = s'_1w_2, s_0 = s'_0w_2$
7	Compute $l(x) = su_1 + v_1 = s_1x^3 + l_2x^2 + l_1x + l_0$
	$l_2 = s_1u_{11} + s_0, l_0 = s_0u_{10} + v_{10}$
	$l_1 = (s_1 + s_0)(u_{11} + u_{10}) - s_1u_{11} - s_0u_{10} + v_{11}$
8	Compute $u' = monic(\frac{F - l^2 - h \cdot l}{u_1^2}) = x^2 + u_{31}x + u_{30}$
	$u_{30} = w_3 \cdot (2v_{11} + h_1 - h_2u_{11} + w_3 \cdot (2u_{11} - f_4 + h_2s_0 + s_0^2))$
	$u_{31} = w_3 \cdot (2s_0 + h_2 - w_3)$
9	Compute $v_3 = -l - h \pmod{u_3} = v_{31}x + v_{30}$
	$w_1 = u_{31}s_1, w_2 = l_2 + h_2 - w_1, w_3 = u_{30}w_2$
	$v_{31} = (u_{31} + u_{30})(w_2 + s_1) - w_3 - w_1 - l_1 - h_1, v_{30} = w_3 - l_0 - h_0$
Cost	23M, 5S, 1I
6'	Compute $l(x) = s_0u_1 + v_1$
	$inv = 1/r, s_0 = s'_0 inv, l_1 = s_0u_{11} + v_{11}, l_0 = s_0u_{10} + v_{10}$
7'	Compute $u_3 = monic(\frac{F - l^2 - h \cdot l}{u_1^2}) = x + u_{30}$
	$u_{30} = f_4 - 2u_{11} - s_0^2 - h_2s_0$
8'	Compute $v_3 = -l - h \pmod{u_3} = v_{30}$
	$v_{30} = u_{30}(l_1 + h_1 - u_{30} \cdot (s_0 + h_2)) - l_0 - h_0$
Cost	14M, 4S, 1I

A General Expansion Method Using Efficient Endomorphisms[⋆]

Tae-Jun Park[1], Mun-Kyu Lee[2], E-yong Kim[1], and Kunsoo Park[1]

[1] School of Computer Science and Engineering,
Seoul National University, Seoul, 151-744, Korea
{tjpark,eykim,kpark}@theory.snu.ac.kr

[2] Electronics and Telecommunications Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
mklee@etri.re.kr

Abstract. There are various expansion methods to accelerate scalar multiplication on special types of elliptic curves. In this paper we present a general expansion method that uses efficient endomorphisms. We first show that the set of all endomorphisms over a non-supersingular elliptic curve E is isomorphic to $\mathbb{Z}[\omega] = \{a + b\omega \mid a, b \in \mathbb{Z}\}$, where ω is an algebraic integer with the smallest norm in an imaginary quadratic field, if ω is an endomorphism over E. Then we present a new division algorithm in $\mathbb{Z}[\omega]$, by which an integer k can be expanded by the Frobenius endomorphism and ω. If ω is more efficient than a point doubling, we can use it to improve the performance of scalar multiplication by replacing some point doublings with the ω maps. As an instance of this general method, we give a new expansion method using the efficiently computable endomorphisms used by Ciet et al. [1].

Keywords: Elliptic Curve, Scalar Multiplication, Frobenius Expansion, Efficient Endomorphism.

1 Introduction

In elliptic curve cryptosystems, scalar multiplication is the most time-consuming operation. To speed up scalar multiplication, various methods that use special curves have been proposed. Since the use of anomalous elliptic curves and Frobenius endomorphisms was suggested in [2] and [3], there have been many Frobenius expansion methods applicable to curves defined over finite fields of characteristic two [4, 5, 6, 7].

Recently, Smart [8] showed that it is possible to use Frobenius expansions on elliptic curves over finite fields of odd characteristic. Kobayashi et al. [9, 10] and Lim and Hwang [11] proposed efficient scalar multiplication algorithms using Smart's expansion method. Gallant, Lambert and Vanstone [12] suggested that efficiently computable endomorphisms other than Frobenius maps can be used for fast scalar multiplication on curves over large prime fields. Park, Lee

[⋆] This work was supported by the MOST grant M6-0203-00-0039.

and Park [13] proposed a new expansion method that combines the Frobenius map and the efficiently computable endomorphisms used in [12]. Ciet et al. [1] proposed another expansion method that uses efficient endomorphisms of norm 2.

In this paper, we present a general expansion method that uses efficient endomorphisms. More specifically, our contributions are as follows.

– We first show that the set of all endomorphisms over a non-supersingular elliptic curve E is isomorphic to $\mathbb{Z}[\omega] = \{a + b\omega \mid a, b \in \mathbb{Z}\}$, where ω is an algebraic integer with the smallest norm in an imaginary quadratic field $\mathbb{Q}(\sqrt{m})$, if ω is an endomorphism over E. Hence, we can use $\mathbb{Z}[\omega]$ to handle endomorphisms over E.
– We present a new division algorithm in $\mathbb{Z}[\omega]$, by which an integer k can be expanded as $k = \sum_{i=0}^{l}(r_{1i} + r_{2i}\omega)\varphi^i$, where r_{1i} and r_{2i} are integers and φ is the Frobenius endomorphism. Hence, if only ω is more efficient than a point doubling, we can use it to improve the performance of scalar multiplication by replacing some point doublings with the ω maps.
– To make a correct comparison between the costs of point doubling and computing endomorphisms, in our implementation we transformed the underlying elliptic curves to have the same form.

As an instance of this general method, we give a new expansion method using the efficiently computable endomorphisms used in [1]. Our experimental results show that the throughputs of scalar multiplications are increased by 3.0–10.4%, when the computations are done on several elliptic curves over optimal extension fields.

2 Preliminaries

2.1 Frobenius Expansions

Let $\mathbb{F}_q$ denote the finite field containing q elements. An elliptic curve over $\mathbb{F}_q$ is given by a Weierstrass equation in the affine coordinate,

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ . \tag{1}$$

$E(\mathbb{F}_q)$ is the set of all pairs $(x, y) \in \mathbb{F}_q \times \mathbb{F}_q$ which satisfy (1), together with the point at infinity $\mathcal{O}$.

Frobenius map φ is given by $\varphi : (x, y) \longrightarrow (x^q, y^q)$ and $\mathcal{O} \longrightarrow \mathcal{O}$, which satisfies the minimal equation

$$\varphi^2 - \tau\varphi + q = 0 \ ,$$

where τ is the trace of φ.

Smart [8] showed that the multiplication-by-k map on $E(\mathbb{F}_{q^n})$ can be expanded in terms of a polynomial in φ if q is odd:

$$k = a_0 + a_1\varphi + \cdots + a_l\varphi^l \ ,$$

where $a_i \in \{-(q+1)/2, \ldots, (q+1)/2\}$. Park et al. [13] proposed new Frobenius expansion methods with efficiently computable endomorphisms whose norms are 1 as follows:

$$k = \sum_{i=0}^{l} r_i \varphi^i \ ,$$

where the coefficients r_i is of the form $r_i = r_{1i} + r_{2i}\omega$ ($r_{1i}, r_{2i} \in \mathbb{Z}$ and ω is an efficiently computable endomorphism whose norm is 1).

2.2 Endomorphism Rings and Imaginary Quadratic Fields

Let $End(E)$ denote the set of all endomorphisms over E. The set $End(E)$ is a ring with two binary operations $(+, \circ)$, where the multiplication is given by composition:

$$(\phi + \psi)(P) = \phi(P) + \psi(P) \ ,$$
$$\psi \circ \phi(P) = \psi(\phi(P)) \ .$$

If E is non-supersingular, $End(E)$ is an order in an imaginary quadratic field $\mathbb{Q}(\sqrt{\tau^2 - 4q})$ [14]. We will investigate the algebraic structure of $End(E)$ more closely.

Let us consider algebraic integers in imaginary quadratic fields. $\mathbb{Q}(\sqrt{m}) = \{u + v\sqrt{m} \mid u, v \in \mathbb{Q}\}$ is an *imaginary quadratic field* if m is a square-free negative integer. An element α in $\mathbb{Q}(\sqrt{m})$ is called an *algebraic integer* if α satisfies a monic polynomial of degree two:

$$x^2 - ax + b = 0 \quad (a, b \in \mathbb{Z}) \ .$$

The *norm* of α is defined by $N(\alpha) = \alpha \cdot \bar{\alpha}$, where $\bar{\alpha}$ is the conjugate of α.

Let A_m denote the ring of all algebraic integers in $\mathbb{Q}(\sqrt{m})$. In [15], it was shown that A_m is as follows.

Theorem 1. [15] *If $m \equiv 2, 3 \pmod 4$, then $A_m = \{a \ + b\sqrt{m} \mid a, b \in \mathbb{Z}\}$. If $m \equiv 1 \pmod 4$, then $A_m = \{\frac{a+b\sqrt{m}}{2} \mid a, b \in \mathbb{Z}, \equiv b \pmod 2\}$.*

Table 1 shows the algebraic integers with the smallest norm for each quadratic field (see [16]).

Table 1. The algebraic integers with the smallest norm for each quadratic field

Field	Smallest norm	Elements of smallest norm
$\mathbb{Q}(\sqrt{m})$, $m \equiv 2,\ 3 \pmod 4$	$-m$	$\pm\sqrt{m}$
$\mathbb{Q}(\sqrt{m})$, $m \equiv 1 \pmod 4$	$\frac{-m+1}{4}$	$\frac{1\pm\sqrt{m}}{2}$, $\frac{-1\pm\sqrt{m}}{2}$

Lemma 1. *If ω is an algebraic integer with the smallest norm in $\mathbb{Q}(\sqrt{m})$, then $A_m = \mathbb{Z}[\omega]$, where $\mathbb{Z}[\omega] = \{a + b\omega \mid a, b \in \mathbb{Z}\}$.*

Proof. If $m \equiv 1 \pmod 4$, the elements of smallest norm are $\frac{1\pm\sqrt{m}}{2}$ and $\frac{-1\pm\sqrt{m}}{2}$. Let $\omega = \frac{1+\sqrt{m}}{2}$. (The other cases are similar.) To prove $A_m \subseteq \mathbb{Z}[\omega]$, consider an element $\frac{x+y\sqrt{m}}{2}$ in A_m. Since $\frac{x+y\sqrt{m}}{2} = \frac{x-y}{2} + y\omega$ and $\frac{x-y}{2}$ is an integer by $x \equiv y \pmod 2$, $\frac{x+y\sqrt{m}}{2}$ is in $\mathbb{Z}[\omega]$. Conversely, since $x + y\omega = \frac{2x+y+y\sqrt{m}}{2}$ and $2x+y \equiv y \pmod 2$, $x+y\omega$ is in A_m. The proof for the case of $m \equiv 2, 3 \pmod 4$ is similar but simpler. □

We turn our attention to the relation between $End(E)$ of non-supersingular E and $\mathbb{Z}[\omega]$.

Theorem 2. *Suppose that there exists an endomorphism ω with the smallest norm in $End(E)$. Then,*

$$End(E) \cong \mathbb{Z}[\omega] \ .$$

Proof. Let $\mathbb{Q}(\sqrt{m})$ be an imaginary quadratic field with ring of algebraic integers A_m. By Silverman [14], the orders of $\mathbb{Q}(\sqrt{m})$ are precisely the rings $\mathbb{Z}+fA_m$ for integers $f > 0$. Since $End(E)$ is an order of $\mathbb{Q}(\sqrt{m})$ for some $m < 0$ and $\mathbb{Z}[\omega]$ is the ring of algebraic integers by Lemma 1, $End(E)$ is of the form $\mathbb{Z} + f\mathbb{Z}[\omega]$. By assumption, $\omega \in End(E) \cong \mathbb{Z} + f\mathbb{Z}[\omega]$. Thus, $f = 1$. We conclude that $End(E) \cong \mathbb{Z}[\omega]$. □

By Theorem 2, the Frobenius map $\varphi \in End(E)$ can be represented by $a + b\omega$ and its norm is represented by the quadratic form $a^2 - mb^2$ if $m \equiv 2, 3 \pmod 4$, or $a^2 \pm ab + \frac{1-m}{4}b^2$ if $m \equiv 1 \pmod 4$.

There are several examples of the curves whose $End(E)$ contains an efficiently computable endomorphism with the smallest norm [12].

Example 1. [17] Let $p \equiv 1 \pmod 4$ be a prime. Consider the elliptic curve

$$E_1 : y^2 = x^3 + ax \tag{2}$$

defined over $\mathbb{F}_p$. Let $\alpha \in \mathbb{F}_p$ be an element of order 4. Then we get an efficiently computable map on $E_1(\mathbb{F}_{p^n})$

$$\lambda : (x, y) \longrightarrow (-x, \alpha y) \quad \text{and} \quad \mathcal{O} \longrightarrow \mathcal{O} \ . \tag{3}$$

If $P \in E_1(\mathbb{F}_{p^n})$ is a point of prime order N, then λ acts on $\langle P \rangle$ as a multiplication map, i.e., $\lambda(Q) = lQ$ for all $Q \in \langle P \rangle$, where l is an integer satisfying $l^2 \equiv -1 \pmod N$. $\lambda(Q)$ can be computed using only n multiplications on $\mathbb{F}_p$.

Note that λ is equivalent to one of $\pm\sqrt{-1}$ (i.e., $m = -1$). Since $N(\lambda) = 1$, λ is one of the endomorphisms with smallest norm 1 as in Table 1. By Theorem 2, $End(E_1) \cong \mathbb{Z}[\lambda]$.

Example 2. [17] Let $p \equiv 1 \pmod 3$ be a prime. Consider the elliptic curve

$$E_2 : y^2 = x^3 + b \tag{4}$$

defined over $\mathbb{F}_p$. Let $\beta \in \mathbb{F}_p$ be an element of order 3. Then we get an efficiently computable map on $E_2(\mathbb{F}_{p^n})$

$$\rho : (x, y) \longrightarrow (\beta x, y) \quad \text{and} \quad \mathcal{O} \longrightarrow \mathcal{O} \ . \tag{5}$$

If $P \in E_2(\mathbb{F}_{p^n})$ is a point of prime order N, then ρ acts on $\langle P \rangle$ as a multiplication map, i.e., $\rho(Q) = kQ$ for all $Q \in \langle P \rangle$, where k is an integer satisfying $k^2 + k \equiv -1 \pmod N$. $\rho(Q)$ can be computed using only n multiplications on $\mathbb{F}_p$.

Note that ρ is equivalent to one of $\frac{-1 \pm \sqrt{-3}}{2}$ (i.e., $m = -3$). Since $N(\rho) = 1$, ρ is one of the endomorphisms with smallest orm 1 as in Table 1. By Theorem 2, $End(E_2) \cong \mathbb{Z}[\rho]$.

Example 3. [17, 1] Let $p > 3$ be a prime such that -7 is a quadratic residue modulo p. Consider the elliptic curve

$$E_3 : y^2 = x^3 - \frac{3}{4}x^2 - 2x - 1 \tag{6}$$

defined over $\mathbb{F}_p$. If $\xi = (1 + \sqrt{-7})/2$ and $a = (\xi - 3)/4$, then the map γ defined in the affine plane by

$$\gamma : (x, y) \longrightarrow \left(\frac{x^2 - \xi}{\xi^2 (x - a)}, \frac{y(x^2 - 2ax + \xi)}{\xi^3 (x - a)^2} \right) \quad \text{and} \quad \mathcal{O} \longrightarrow \mathcal{O} \tag{7}$$

is an endomorphism on $E_3(\mathbb{F}_{p^n})$. Moreover, γ satisfies the equation $\gamma^2 - \gamma + 2 = 0$. In affine coordinates, computing γ is more expensive than a point doubling. In projective coordinates, however, if we count the number of operations on $\mathbb{F}_{p^n}$, γ can be computed using only 5 multiplications and 2 squarings while point doubling needs 7 multiplications and 5 squarings. (See Section 4.1 for more details.)

Note that γ is equivalent to one of $\frac{1 \pm \sqrt{-7}}{2}$ (i.e., $m = -7$). Since $N(\gamma) = 2$, γ is one of the endomorphisms with smallest norm 2 as in Table 1. By Theorem 2, $End(E_3) \cong \mathbb{Z}[\gamma]$.

Example 4. [18, 1] Let $p > 3$ be a prime such that -2 is a quadratic residue modulo p. Consider the elliptic curve

$$E_4 : y^2 = 4x^3 - 30x - 28 \tag{8}$$

defined over $\mathbb{F}_p$. The map δ defined in the affine plane by

$$\delta : (x, y) \longrightarrow \left(-\frac{2x^2 + 4x + 9}{4(x + 2)}, -\frac{2x^2 + 8x - 1}{4\sqrt{-2}(x + 2)^2} y \right) \quad \text{and} \quad \mathcal{O} \longrightarrow \mathcal{O} \tag{9}$$

is an endomorphism on $E_4(\mathbb{F}_{p^n})$. Moreover, δ satisfies the equation $\delta^2 + 2 = 0$. In affine coordinates, computing δ is more expensive than a point doubling. In projective coordinates, however, if we count the number of operations on $\mathbb{F}_{p^n}$, δ can be computed using only 5 multiplications and 2 squarings. (See Section 4.1 for more details.)

Note that δ is equivalent to one of $\pm\sqrt{-2}$ (i.e., $m = -2$). Since $N(\delta) = 2$, δ is one of the endomorphisms with smallest norm 2 as in Table 1. By Theorem 2, $End(E_4) \cong \mathbb{Z}[\delta]$.

3 General Methods for New Frobenius Expansions

3.1 Case of $m \equiv 2,\ 3 \pmod 4$

In this section, we present a new expansion method for the case that $m \equiv 2,\ 3 \pmod 4$ and there exists $\omega \in End(E)$ whose norm is $-m$. We begin by giving a new division algorithm. Let $\varphi = a\ + b\omega$ whose norm is prime $p = a^2 - mb^2$. For any $s \in \mathbb{Z}[\omega]$, we can find $t,\ r \in \mathbb{Z}[\omega]$ such that $s = t\varphi + r$ by Algorithm 1.

Algorithm 1 New division algorithm for $m \equiv 2,\ 3 \pmod 4$

```
Input: $\varphi = a\ + b\omega,\ s = s_1 + s_2\omega \in \mathbb{Z}[\omega]$.
Output: $t = t_1 + t_2\omega,\ r = r_1 + r_2\omega$.
  begin
    step 1 $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leftarrow \frac{1}{p}\begin{pmatrix} a & -mb \\ -b & a \end{pmatrix}\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$
    step 2 $\begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \leftarrow \begin{pmatrix} \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \end{pmatrix}$, where $\lfloor z \rceil$ means the nearest integer to $z$.
    step 3 $\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \leftarrow \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} - \begin{pmatrix} a & mb \\ b & a \end{pmatrix}\begin{pmatrix} \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \end{pmatrix}$
  end
```

We can show that $\|r\| \le \frac{\sqrt{p(1-m)}}{2}$, where $\|\cdot\| = \sqrt{N(\cdot)}$. See Fig. 1. Since $s, t, \varphi \in \mathbb{Z}[\omega]$, s and $t\varphi$ are in the 2-dimensional integer lattice L_1 generated by 1 and ω. Note that $t\varphi$ is also in the integer lattice L_2 generated by φ and $\omega\varphi$, but s is not. Thus, computing r by Algorithm 1 is equivalent to finding a point in L_2 nearest to s. See the parallelogram in Fig. 1. Since the absolute values between the center point and vertices of the parallelogram are $\le \frac{\sqrt{p(1-m)}}{2}$, we can easily see that $\|r\| \le \frac{\sqrt{p(1-m)}}{2}$. It is also easy to prove that the number of such remainders r for the new division algorithm is exactly p.

Using the new division algorithm, we can expand any s by φ as Smart's expansion method does. We prove below that the expansion length is finite. Note that $\varphi = a\ + b\omega$, where $a,\ b \neq 0$, and p is odd prime.

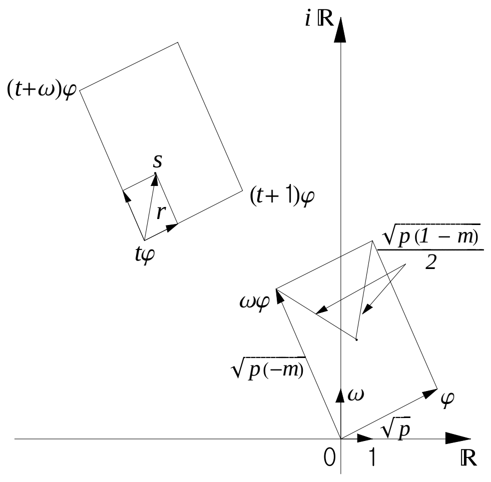


Fig. 1. Computing t and r given s

Theorem 3. *We can write for* $s \in \mathbb{Z}[\omega]$,

$$s = \sum_{i=0}^{l} r_i \varphi^i \ ,$$

where $\|r_i\| \le \frac{\sqrt{p(1-m)}}{2}$ *and* $l \le \lceil 2 \log_p \|s\| \rceil + 2$ *with the following exception.*

$m = -2$	$p = 3$	$l \le \lceil 2 \log_p \|s\| \rceil + 3$

Proof. Omitted.

3.2 Case of $m \equiv 1 \pmod 4$

In this section, we present a new expansion method for the case that $m \equiv 1 \pmod 4$ and there exists $\omega \in End(E)$ whose norm is $\frac{1-m}{4}$. We begin by giving a new division algorithm. Let $\varphi = a \ + b\omega$ whose norm is prime $p = a^2 \pm ab + \frac{1-m}{4} b^2$. For any $s \in \mathbb{Z}[\omega]$, we can find $t,\ r \in \mathbb{Z}[\omega]$ such that $s = t\varphi + r$ by Algorithm 2.

We can prove that $\|r\| \le \frac{\sqrt{p(9-m)}}{4}$. See Fig. 2. Since $s, t, \varphi \in \mathbb{Z}[\omega]$, s and $t\varphi$ are in the 2-dimensional integer lattice L_1 generated by 1 and ω. Note that $t\varphi$ is also in the integer lattice L_2 generated by φ and $\omega\varphi$, but s is not. Thus, computing r by Algorithm 2 is equivalent to finding a point in L_2 nearest to s. See the parallelogram in Fig. 2. Since the absolute values between the center point and vertices of the parallelogram are $\le \frac{\sqrt{p(9-m)}}{4}$, we can easily see that $\|r\| \le \frac{\sqrt{p(9-m)}}{4}$. The number of remainders r for the new division algorithm is exactly p.

Algorithm 2 New division algorithm for $m \equiv 1 \pmod 4$

```
Input: $\varphi = a + b\omega,\ s = s_1 + s_2\omega \in \mathbb{Z}[\omega]$.
Output: $t = t_1 + t_2\omega,\ r = r_1 + r_2\omega$.
  begin
    step 1 $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leftarrow \frac{1}{p}\begin{pmatrix} a \pm b & \frac{1-m}{4}b \\ -b & a \end{pmatrix}\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$
    step 2 $\begin{pmatrix} t_1 \\ t_2 \end{pmatrix} \leftarrow \begin{pmatrix} \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \end{pmatrix}$
    step 3 $\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \leftarrow \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} - \begin{pmatrix} a & -\frac{1-m}{4}b \\ b & a \pm b \end{pmatrix}\begin{pmatrix} \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \end{pmatrix}$
  end
```

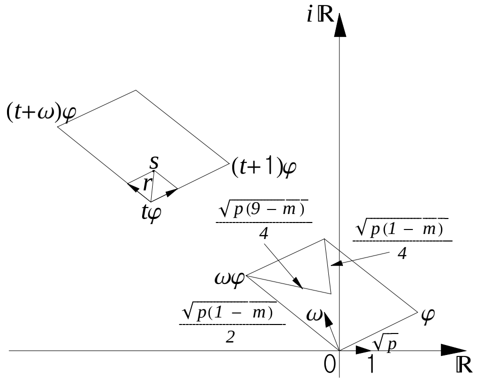


Fig. 2. Computing t and r given s

Using the new division algorithm, we can expand any s with φ. We prove below that the expansion length is finite. Note that $\varphi = a + b\omega$, where $a,\ b \neq 0$, and p is odd prime.

Theorem 4. *We can write for $s \in \mathbb{Z}[\omega]$,*

$$s = \sum_{i=0}^{l} r_i \varphi^i \ ,$$

where $\|r_i\| \leq \frac{\sqrt{p(9-m)}}{4}$ and $l \leq \lceil 2\log_p \|s\| \rceil + 3$ with the following exceptions.

$m = -3$	$p = 3$	*may be periodic infinite*
$m = -11$	$p = 3$ $p = 5$	$l \leq \lceil 2\log_p \|s\| \rceil + 1$

Proof. Omitted.

4 Implementations

4.1 Using Projective Coordinates

An elliptic curve can be represented by several coordinate systems. In projective coordinates, a projective point (X, Y, Z) corresponds to the affine point $(X/Z, Y/Z)$. In Jacobian coordinates, a projective point (X, Y, Z) corresponds to the affine point $(X/Z^2, Y/Z^3)$. Here, we choose projective coordinates since the number of operations needed to compute γ in (7) or δ in (9) is minimal and the point additions are cheaper than in Jacobian coordinates.

We will show that in curves (6) and (8), computing γ and δ can be faster than point doubling. In affine coordinates, computing γ (or δ) is more expensive than point doubling [12, 1]. However, in projective coordinates, γ can be computed as

$$\begin{aligned} \gamma(X,Y,Z) &= (EF, Y(A-2XD+C), F^2B) \ , \\ A = X^2 \ , \quad B &= \xi Z \ , \quad C = BZ \ , \quad D = aZ \ , \\ E = A - C \ , &\quad F = (X-D)\xi \ , \end{aligned} \tag{10}$$

and δ can be computed as

$$\begin{aligned} \delta(X,Y,Z) &= (D(2A+4B+9Z^2), (2A+8B-Z^2)Y, -4DCZ) \ , \\ A = X^2 \ , \quad B &= XZ \ , \quad C = X+2Z \ , \quad D = \sqrt{-2}C \ . \end{aligned} \tag{11}$$

Therefore, computing γ is comparable to a point doubling, and computing δ is significantly faster than a point doubling on the prime fields $\mathbb{F}_p$ [1]. Moreover, if we use the extension fields $\mathbb{F}_{p^n}$ for the underlying fields, we can ignore the costs of subfield operations such as the multiplication by ξ, a in (10), and $\sqrt{-2}$ in (11). Hence, we can decrease the number of operations in computing γ and δ. We summarize the number of operations needed by each point operation in Table 2.

However, the number of operations for doubling a point is counted on the elliptic curve of the form $y^2 = x^3 + ax + b$, not of the form (6) or (8) [1, 19]. Hence, in order to make a correct comparison between point doubling and computing γ or δ, we have to either transform the curves to the form having the above costs

Table 2. Number of operations in projective coordinates

	Point doubling	Computing γ	Computing δ
Multiplication	7	$5^{\dagger}$ $(8^{\ddagger})$	$5^{\dagger}$ $(6^{\ddagger})$
Squaring	5	2	2

† Counted on the extension fields $\mathbb{F}_{p^n}$
‡ Counted on the prime fields $\mathbb{F}_p$ [1]

$$\begin{tikzcd} \underset{(X,Y,Z)}{E_3} \arrow[r, "T_1"] \arrow[d, "\gamma"'] & \underset{(4X-Z,4Y,4Z)}{\overline{E}_3} \arrow[d, "\overline{\gamma}"] \\ \underset{\gamma(X,Y,Z)}{E_3} \arrow[r, "T_1"] & \underset{\overline{\gamma}(4X-Z,4Y,4Z)}{\overline{E}_3} \end{tikzcd} \qquad \begin{tikzcd} \underset{(X,Y,Z)}{E_4} \arrow[r, "T_2"] \arrow[d, "\delta"'] & \underset{(2X,Y,2Z)}{\overline{E}_4} \arrow[d, "\delta"] \\ \underset{\delta(X,Y,Z)}{E_4} \arrow[r, "T_2"] & \underset{\delta(2X,Y,2Z)}{\overline{E}_4} \end{tikzcd}$$

Fig. 3. Commutative diagrams with projective points

or make new formulas for point doubling. Here, we choose the former since new formulas do not guarantee to have the same costs as in Table 2.

We transform E_3 to another form by using the admissible change of variables $T_1 : (x, y) \to (x - \frac{1}{4}, y)$. Then, we have the transformed elliptic curve

$$\overline{E}_3 : y^2 = x^3 - \frac{35}{16}x - \frac{49}{32} ,$$

and the endomorphism γ is changed to a somewhat different form

$$\begin{aligned} \overline{\gamma}(X,Y,Z) &= (4EF - F^2B, 16Y(A - 2(4X+Z)D + C), 4F^2B) , \\ A &= (4X+Z)^2 , \quad B = 4\xi Z , \quad C = 4BZ , \quad D = 4aZ , \\ E &= A - C , \quad F = (4X + Z - D)\xi , \end{aligned}$$

compared with the formulas in (10). Note that the operation costs are not changed. Now, we can use the doubling formulas in [19] on this curve.

Similarly, we transform E_4 to another form by using the admissible change of variables $T_2 : (x, y) \to (x, \frac{y}{2})$. Then, we have the transformed elliptic curve

$$\overline{E}_4 : y^2 = x^3 - \frac{15}{2}x - 7 .$$

Note that the endomorphism δ is not changed on the transformed curve $\overline{E}_4$, so we use the same formulas as in (11). As in the case of γ, we can use the doubling formulas in [19] on this curve.

Note that the endomorphisms are still efficiently computable on the transformed curves, since the operation costs remain as in Table 2. Hence, we can compare the speed of computing the endomorphism $\overline{\gamma}$ (δ) to that of doubling a point on the curve $\overline{E}_3$ ($\overline{E}_4$) instead of E_3 (E_4). The related commutative diagrams are shown in Fig. 3.

4.2 Scalar Multiplication Algorithms

In this section, we describe the scalar multiplication algorithm using our general expansion method. First, we explain briefly the Kobayashi-Morita-Kobayashi-Hoshino (KMKH) algorithm ([9, 10]) which uses only Frobenius endomorphism to expand k for the computation of kP. Then we present the modified algorithm which uses our general expansion method. Finally, we compare the number of point operations between the two algorithms.

Original algorithm We briefly describe the KMKH algorithm. First, the algorithm expands k as

$$k = \sum_{i=0}^{l} u_i \varphi^i \ ,$$

where $-\frac{p}{2} \leq u_i \leq \frac{p}{2}$ and $l = \lceil 2 \log_p k \rceil + 3 \simeq 2n + 3$ by using $\varphi^2 - \tau\varphi + p = 0$. Next, it reduces the expansion length using $\varphi^n(P) = P$, so k is represented as

$$k = \sum_{i=0}^{n-1} r_i \varphi^i \ .$$

Then, a simultaneous scalar multiplication is done to compute the sum of individual scalar multiple $r_i P_i$, where $P_i = \varphi^i(P)$.

Improved algorithm We show the general scalar multiplication algorithm ωKMKH which uses our general expansion method. The map ω can be any efficient endomorphism. In the pseudo-codes, the first subscripts denote array indices, and the second subscripts denote bit positions, where the least significant bit is regarded as the 0-th bit. The whole procedure is similar to the original KMKH algorithm. In line 3, the matrix A is different for each endomorphism. (See Table 3.) In line 10, we make the precomputation table according to each endomorphism.

Algorithm 3 consists of three steps. Lines 1–4 are the first step, in which k is expanded using both φ and ω. Lines 5–7 are the second step, optimization of the expansion length using $\varphi^n(P) = P$. We use NAF for representing r_{1_i}'s and r_{2_i}'s for speedup. Next, we make the precomputation tables P_1 and P_2 in lines 8–10. Finally, we do parallel scalar multiplication in lines 11–18.

Table 3. Matrices for each endomorphism in Algorithm 3, line 3 ($\varphi = a \ + b\omega$)

Endomorphism ω	λ	ρ	$\gamma(\overline{\gamma})$	δ
Matrix A	$\begin{pmatrix} a & -b \\ b & a \end{pmatrix}$	$\begin{pmatrix} a & -b \\ b & a - b \end{pmatrix}$	$\begin{pmatrix} a & -2b \\ b & a + b \end{pmatrix}$	$\begin{pmatrix} a & -2b \\ b & a \end{pmatrix}$
Matrix A^{-1}	$\frac{1}{p}\begin{pmatrix} a & b \\ -b & a \end{pmatrix}$	$\frac{1}{p}\begin{pmatrix} a - b & b \\ -b & a \end{pmatrix}$	$\frac{1}{p}\begin{pmatrix} a + b & 2b \\ -b & a \end{pmatrix}$	$\frac{1}{p}\begin{pmatrix} a & 2b \\ -b & a \end{pmatrix}$

Algorithm 3 General ωKMKH procedure

```
Input: $k, P$
Output: $Q = kP$
1: $i \leftarrow 0$,  $s_1 \leftarrow k$, $s_2 \leftarrow 0$,  $u_{1_j} \leftarrow 0$, $u_{2_j} \leftarrow 0$  for $0 \le j < 3n$.
2: while $s_1 \neq 0$ or $s_2 \neq 0$ do
3:   $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leftarrow A^{-1}\begin{pmatrix} s_1 \\ s_2 \end{pmatrix}$,  $\begin{pmatrix} u_{1_i} \\ u_{2_i} \end{pmatrix} \leftarrow \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} - A\begin{pmatrix} \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \end{pmatrix}$,  $\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \leftarrow \begin{pmatrix} \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \end{pmatrix}$.
4:   $i \leftarrow i + 1$.
5: for $i = 0$ to $n - 1$ do
6:   $d_{1_i} \leftarrow u_{1_i} + u_{1_{(i+n)}} + u_{1_{(i+2n)}}$,  $r_{1i} \leftarrow \mathrm{NAF}(d_{1_i})$.
7:   $d_{2_i} \leftarrow u_{2_i} + u_{2_{(i+n)}} + u_{2_{(i+2n)}}$,  $r_{2_i} \leftarrow \mathrm{NAF}(d_{2_i})$.
8: for $i = 0$ to $n - 1$ do
9:   $P_{1_i} \leftarrow \varphi^i P$.
10:  $P_{2_i} \leftarrow \omega(\varphi^i(P))$.
11: $Q \leftarrow \mathcal{O}$.
12: for $j = \lceil (\log_2 p)/2 \rceil + 1$ downto 0 do
13:   $Q \leftarrow 2Q$.
14:   for $i = 0$ to $n - 1$ do
15:     if $(r_{1_{ij}} = 1)$ then  $Q \leftarrow Q + P_{1_i}$.
16:     else if $(r_{1_{ij}} = -1)$ then  $Q \leftarrow Q - P_{1_i}$.
17:     if $(r_{2_{ij}} = 1)$ then  $Q \leftarrow Q + P_{2_i}$.
18:     else if $(r_{2_{ij}} = -1)$ then  $Q \leftarrow Q - P_{2_i}$.
```

Comparison of The Number of Operations We compare the number of point operations between KMKH and ωKMKH. (See Table 4.) The number of coefficients is doubled but the size of each coefficient is reduced by a square root order in ωKMKH. Hence, the number of point doublings is reduced to a half, but ωKMKH computes the ω map n times. The number of point additions is almost the same. In other words, ωKMKH computes less number of point doublings than KMKH, but computes the ω map for precomputing $\omega(\varphi^i(P))$'s instead. In cases of λ and ρ, ωKMKH is faster than KMKH since computing λ and ρ is almost free. In cases of $\overline{\gamma}$ and δ, however, computing $\overline{\gamma}$ and δ is comparable to point doublings. Therefore n needs to be less than $(\log_2 p)/2$.

Table 4. Comparison of the number of point operations

	KMKH	ωKMKH
Number of coefficients	n	$2n$
Number of bits in each coefficient	$\log_2 p$	$(\log_2 p)/2$
Number of point doublings	$\log_2 p$	$(\log_2 p)/2$
Number of computing ω	0	n
Average number of point additions	$(n \log_2 p)/2$	$(n \log_2 p)/2$

Table 5. Implemented fields and curves

	p	n	Irreducible binomial	Bitlength of order	Curve, ω
1	$2^{24}+315$	11	$f(x)=x^{11}-2$	221	$\overline{E}_3,\ \overline{\gamma}$
2	$2^{25}+497$	11	$f(x)=x^{11}-2$	228	
3	$2^{27}-39$	11	$f(x)=x^{11}-2$	255	
4	$2^{24}+73$	11	$f(x)=x^{11}-2$	222	$\overline{E}_4,\ \delta$
5	$2^{25}-735$	11	$f(x)=x^{11}-2$	250	
6	$2^{30}+889$	7	$f(x)=x^{7}-2$	161	
7	$2^{31}-589$	7	$f(2)=x^{7}-2$	169	

5 Experiments

In this section, we show experimental results of the algorithms in the previous section. In cases of the endomorphisms λ and ρ, our method improves the throughput by 10.5–16.1% for the KMKH algorithm [13]. We show the results of the endomorphisms $\overline{\gamma}$ and δ here.

For the underlying fields, we consider only optimal extension fields $\mathbb{F}_{p^n}$, where p is selected to fit into a CPU word [20, 21]. The fields and curves which we have implemented are shown in Table 5. We choose projective coordinates as mentioned before.

Table 6 shows the evaluated timings for scalar multiplications with various algorithms. We implemented our algorithms in C++ using gcc-3.2 compiler. The timings are measured on a Pentium4 2.4GHz workstation. In the KMKH algorithm, r_i's in (4.2) are represented by NAF as r_{1i}'s and r_{2i}'s in the ωKMKH algorithm. Our method improves the throughput by 3.0–10.4% upon the KMKH algorithm.

Table 6. Timings for point operations and scalar multiplications

	Curve, ω	Point operations (μsec)		Scalar multiplications (msec)		Gain[†]
		Point doubling	Computing ω	KMKH	ωKMKH	
1	$\overline{E}_3,\ \overline{\gamma}$	265.26	170.63	34.15	32.66	4.5%
2		264.32	172.15	35.29	34.25	3.0%
3		276.20	171.66	38.80	37.21	4.3%
4	$\overline{E}_4,\ \delta$	260.56	166.22	33.72	32.18	4.8%
5		273.68	171.60	37.46	34.95	7.2%
6		136.58	84.59	12.98	11.79	10.1%
7		122.04	79.40	13.76	12.46	10.4%

[†] Throughput increase of ωKMKH over KMKH

Table 6 also shows the timings for point doubling and computing ω. Note that computing $\overline{\gamma}$ and δ is faster than a point doubling. In cases of λ and ρ, the costs of computing them are almost free, but computing $\overline{\gamma}$ and δ is quite comparable to point doublings. So, the gains are somewhat decreased compared with those of λ and ρ.

6 Conclusions

We proposed a general expansion method that uses the Frobenius endomorphism and other efficient endomorphisms defined on an elliptic curve over an odd characteristic field. By replacing some of the doublings in a scalar multiplication with more efficient maps, our method improves the scalar multiplication algorithms that use Frobenius expansion.

Our method requires that p satisfy a specific condition and it uses special curves. Hence the number of suitable curves is smaller than those of the original Frobenius expansion methods. However, it does not seem to be a problem, since there exist many curves that are suitable for cryptographic use (i.e., curves that have a large prime factor in their group orders). Furthermore, there can be other efficient endomorphisms than the ones listed in this paper. It would be interesting to find such endomorphisms.

Finally, we remark that there is no known attack that significantly reduces the time required to compute elliptic curve discrete logarithms on curves such as ones used in this paper [12].

References

[1] Ciet, M., Lange, T., Sica, F., Quisquater, J. J.: Improved algorithms for efficient arithmetic on elliptic curves using fast endomorphisms. In: Advances in Cryptology-EUROCRYPT 2003. Volume 2656 of LNCS., Springer-Verlag (2003) 388–400 112, 113, 116, 120

[2] Koblitz, N.: CM-curves with good cryptographic properties. In: Advances in Cryptology-CRYPTO 91. Volume 576 of LNCS., Springer-Verlag (1991) 279–287 112

[3] Meier, W., Staffelbach, O.: Efficient multiplication on certain non-supersingular elliptic curves. In: Advances in Cryptology-CRYPTO 92. Volume 740 of LNCS., Springer-Verlag (1992) 333–344 112

[4] Müller, V.: Fast multiplication on elliptic curves over small fields of characteristic two. Journal of Cryptology **11** (1998) 219–234 112

[5] Cheon, J., Park, S., Park, S., Kim, D.: Two efficient algorithms for arithmetic of elliptic curves using Frobenius map. In: Public Key Cryptography 98. Volume 1431 of LNCS., Springer-Verlag (1998) 195–202 112

[6] Solinas, J.: An improved algorithm for arithmetic on a family of elliptic curves. In: Advances in Cryptology-CRYPTO 97. Volume 1294 of LNCS., Springer-Verlag (1997) 357–371 112

[7] Solinas, J.: Efficient arithmetic on Koblitz curves. Designs, Codes and Cryptography **19** (2000) 195–249 112

[8] Smart, N.: Elliptic curve cryptosystems over small fields of odd characteristic. Journal of Cryptology **12** (1999) 141–151 112, 113

[9] Kobayashi, T., Morita, H., Kobayashi, K., Hoshino, F.: Fast elliptic curve algorithm combining Frobenius map and table reference to adapt to higher characteristic. In: Advances in Cryptology-EUROCRYPT 99. Volume 1592 of LNCS., Springer-Verlag (1999) 176–189 112, 121

[10] Kobayashi, T.: Base-ϕ method for elliptic curves over OEF. IEICE Trans. Fundamentals **E83-A** (2000) 679–686 112, 121

[11] Lim, C., Hwang, H.: Speeding up elliptic scalar multiplication with precomputation. In: Information Security and Cryptology-ICISC 99. Volume 1787 of LNCS., Springer-Verlag (1999) 102–119 112

[12] Gallant, R., Lambert, R., Vanstone, S.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Advances in Cryptology-CRYPTO 2001. Volume 2139 of LNCS., Springer-Verlag (2001) 190–200 112, 113, 115, 120, 125

[13] Park, T., Lee, M., Park, K.: New frobenius expansions for elliptic curves with efficient endomorphisms. In: ICISC 2002. Volume 2587 of LNCS., Springer-Verlag (2002) 264–282 113, 114, 124

[14] Silverman, J.: The Arithmetic of Elliptic Curves. Springer-Verlag (1986) 114, 115

[15] Hardy, G., Wright, E.: An Introduction to the Theory of Numbers. 3rd edn. Oxford University Press (1954) 114

[16] Gilbert, W.: Radix representations of quadratic fields. J. Math. Anal. Appl. **83** (1981) 264–274 114

[17] Cohen, H.: A Course in Computational Algebraic Number Theory. 3rd edn. Springer-Verlag (1996) 115, 116

[18] Cox, D.: Primes of the Form $x^2 + ny^2$. Fermat, Class Field Theory and Complex Multiplication. Wiley (1998) 116

[19] Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In: Advances in Cryptology-ASIACRYPT 1998. Volume 1514 of LNCS., Springer-Verlag (1998) 51–65 120, 121

[20] Bailey, D., Paar, C.: Optimal extension fields for fast arithmetic in public key algorithms. In: Advances in Cryptology-CRYPTO 98. Volume 1462 of LNCS., Springer-Verlag (1998) 472–485 124

[21] Bailey, D., Paar, C.: Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. Journal of Cryptology **14** (2001) 153–176 124

Design of Bit Parallel Multiplier with Lower Time Complexity

Seon Ok Lee[1], Seok Won Jung[1], Chang Han Kim[2], Janghong Yoon[3], Jae-Young Koh[3], and Daeho Kim[3]

[1] Center for Information Security Technologies(CIST)
Korea University, Seoul, Korea
{seonognim,jsw}@cist.korea.ac.kr
[2] Dept. of Information Security
Semyung Univ.,Jechon, Korea
chkim@semyung.ac.kr
[3] National Security Research Institute(NSRI),
{jhyoon,jykoh,dhokim}@etri.re.kr

Abstract. Recently efficient implementation of finite field operations has received a lot of attention. Among $GF(2^m)$ arithmetic operations, a multiplication process is the most basic and a critical operation that determines a speed-up in hardware. Mastrovito multipliers using a trinomial $p(x) = x^m + x^n + 1 (n \neq m/2)$ require $m^2 - 1$ XOR gates and m^2 AND gates. The proposed multiplier that depends on the intermediate term x^n needs m^2 AND gates and $m^2 + (n^2 - 3n)/2$ XOR gates. The time complexity of existing multipliers is $T_A + (\lfloor (m-2)/(m-n) \rfloor + 1 + \lceil \log_2 m \rceil)\, T_X$ ([12]) and that of the proposed method is $T_A + (1 + \lceil \log_2(m - 1 + \lceil n/2 \rceil) \rceil)\, T_X$. The proposed architecture is efficient for the extension degree m suggested as standards: SEC2, ANSI X9.63. In average, the space complexity is increased to 1.18% but the time complexity is reduced 9.036%.

1 Introduction

Efficient hardware implementations of arithmetic operations in the Galois field $GF(2^m)$ are frequently required in coding theory, computer algebra, and public-key cryptography ([5, 8, 9]). The research of an efficient multiplier distributes an improvement to a multiplicative implementation of a cryptosystem. The measure of an efficiency is the number of gates, XOR gate and AND gate, and the total gate delay of the circuit.

Among arithmetic operations over $GF(2^m)$, a multiplication is the basic operation and an important factor in many applications. Efficient multiplication methods and architectures have been proposed over $GF(2^m)$ in which different basis representations of field elements are used, such as standard basis, dual basis, and normal basis. Standard basis is more promising in the sense that it gives designers freedom on irreducible polynomials selection and hardware optimization. The multiplication in $GF(2^m)$ based on standard basis is often accomplished in two-step algorithms: the polynomial multiplication and the modular reduction. Mastrovito proposed a bit-parallel multiplier combining the above two steps together ([6, 7]). In general cases, the complexity of computing the product matrix is proportional to the Hamming weight of an irreducible polynomial. Sunar and Koç([12]) showed that a general Mastrovito multiplier with an irreducible trinomial $x^m+x^n+1 (n \neq m/2)$ required m^2-1 XOR gates and m^2 AND gates. A new formulation of the Mastrovito multiplication matrix over the field $GF(2^m)$ was generated by an arbitrary irreducible polynomial in [2].

We propose a new architecture design using the Mastrovito's product matrix Z and an irreducible trinomial which is more efficient than the Sunar and Koç's method from the view point of the time complexity. We find a numerical formula of the product matrix which can be decomposed into three matrices: $Z = S+Q+R$ where the matrix S has some entries coming from only matrix M, the matrix Q and matrix R have elements related to the modular reduction. This characteristic suggests that the proposed multiplier constructs a parallel system. As an unit(Block) of the multiplier is arranged efficiently, the time complexity is less than existing multipliers([2, 12, 14]).

This paper is organized as follows: We introduce the method of the Two-Step Algorithm and the Mastrovito multiplication in Section 2. The bit-parallel design of the Mastrovito multiplier architecture using a trinomial are suggested in Section 3. We analyze the time and space complexity in Section 4. We compare the efficiency of a proposed multiplier with that of existing multipliers ([2, 12]) and conclude this study.

2 The Existing Multiplication Methods

In this section, we introduce two multiplication methods which are the two-step method and the Mastrovito method to construct a bit-parallel multiplier over $GF(2^m)$ with the polynomial basis.

2.1 The Two-Step Algorithm

Let $p(x)$ be an irreducible trinomial of the form $x^m + x^n + 1$, where $m > 2n$ without loss of generality ([3]). Let

$$A(x) = \sum_{i=0}^{m-1} a_i x^i, \quad B(x) = \sum_{i=0}^{m-1} b_i x^i,$$

where $a_i, b_i \in GF(2)$.

The multiplication of $A(x)$ and $B(x)$ in $GF(2^m)$ usually consists of two steps, the polynomial multiplication and the modular reduction:

$$C'(x) = A(x) \times B(x),$$
$$C(x) = C'(x) \bmod p(x),$$

where $C'(x)$ is a polynomial of degree at most $2m-2$ and $C(x) \in GF(2^m)$.

■ **The polynomial multiplication:** We first obtain the product polynomial $C'(x)$. It is represented by

$$c' = M \cdot b,$$

where $b = [b_0, b_1, \cdots, b_{m-1}]^T$, $c' = [c'_0, c'_1, \cdots, c'_{2m-2}]^T$ are the coefficient vectors of $B(x)$ and $C'(x)$, respectively, and a $(2m-1) \times m$ matrix M is given as

$$M = \begin{pmatrix} a_0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & \cdots & 0 & 0 \\ \vdots & & \vdots & & \ddots & \vdots & \vdots \\ a_{m-2} & a_{m-3} & a_{m-4} & a_{m-5} & \cdots & a_0 & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_1 & a_0 \\ 0 & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_2 & a_1 \\ \vdots & & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{m-1} \end{pmatrix}.$$

■ **The modular reduction:** The reduction operation is performed in order to obtain $C(x)$,

$$\begin{aligned} C(x) &= C'(x) \mod p(x) \\ &= \sum_{i=0}^{2m-2} c'_i x^i \mod p(x) \\ &= \sum_{i=0}^{m-1} c'_i x^i + \sum_{i=m}^{2m-2} c'_i \Big(x^i \mod p(x) \Big). \end{aligned}$$

Since $x^m = 1 + x^n \mod p(x)$, in case $i \geqq m$, $x^i \mod p(x)$ is divided into two cases as follows:

$$\begin{aligned} x^{m+k} &= x^k + x^{n+k}, & k &= 0, 1, \cdots, m-n-1, \\ x^{2m-n+l} &= x^{m-n+l} + x^l + x^{n+l}, & l &= 0, 1, \cdots, n-2. \end{aligned}$$

Table 1 shows the above equation explicitly.

2.2 Mastrovito Multiplication Algorithm

In this sub-section, we will describe a Mastrovito multiplication with a trinomial $p(x) = x^m + x^n + 1$ as stated in [12]. This method combines the polynomial multiplication step and the modular reduction step of the two step Algorithm.

The multiplication $C(x)$ is obtained by $c = Z \cdot b$, where $c = [c_0, c_1, \cdots, c_{m-2}, c_{m-1}]^T$ is a coefficient vector of $C(x)$. The $m \times m$ matrix Z is called the product

Table 1. The reduction array

x^m	$= x^0$	$+ x^n.$	
x^{m+1}	$= x^1$	$+ x^{n+1}.$	
	$\vdots$		
x^{m+k}	$= x^k$	$+ x^{n+k}.$	
	$\vdots$		
x^{2m-n-1}	$= x^{m-n-1}$	$+ x^{m-1}.$	
x^{2m-n}	$= x^{m-n}$	$+ x^0$	$+ x^n.$
	$\vdots$		
x^{2m-n+l}	$= x^{m-n+l}$	$+ x^l$	$+ x^{n+l}.$
	$\vdots$		
x^{2m-2}	$= x^{m-2}$	$+ x^{n-2}$	$+ x^{2n-2}.$

matrix. In order to explain how to compute the matrix Z, let's define some notations. Let M_i be the i-th row of a matrix M. Let $M[\downarrow i]$ represent a matrix M shifted down i rows by feeding i rows of zeros from top. For example,

$$M[\downarrow 2] = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m-2} & a_{m-3} & a_{m-4} & a_{m-5} & \cdots & a_0 & 0 \\ 0 & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_2 & a_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & a_{m-2} & a_{m-3} \end{pmatrix}.$$

Let $M[\uparrow i]$ represent the matrix M shifted up i rows by feeding i rows of zeros from bottom. Let $M[\rightarrow i]$ represent the matrix M shifted right i columns by feeding i columns of zeros from left.The $m \times m$ sub-matrix X is the upper m rows of M, and the $m \times m$ sub-matrix T is concatenation of the lower $m-1$ rows of M and a zero row:

$$X = \begin{pmatrix} a_0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_0 & 0 & \cdots & 0 & 0 \\ a_2 & a_1 & a_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_0 & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 \end{pmatrix}, \; T = \begin{pmatrix} 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{m-1} & a_{m-2} \\ 0 & 0 & 0 & \cdots & 0 & a_{m-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}.$$

In [12], the product matrix Z is obtained by systematically reducing the last $m-1$ rows of the $(2m-1) \times m$ matrix M with the reduction array, Table 1.

Algorithm 1. Generating a product matrix for $x^m + x^n + 1$, $m > 2n$

```
INPUT: Irreducible trinomial $p(x)$, Matrix $X$, Matrix $T$.
OUTPUT: Matrix $Z$.
 1. $T1 \leftarrow T[\uparrow (m-n)]$;
 2. $U \leftarrow T[\downarrow n]$;
 3. $U1 \leftarrow U[\rightarrow (m-n)]$;
 4. $Y \leftarrow T + T1 + U + U1$;
 5. $Z \leftarrow X + Y$;
 6. Return $(Z)$.
```

The rows of the reduction array can be divided into groups consisting of rows with a relative number of reductions.

The columns of the reduction array have some characteristics. On the right hand side, the first column is sequence of increasing $1, x, x^2, x^3, \cdots, x^{m-2}$. The second column divides into two sequences: The sequence $x^n, x^{n+1}, \cdots, x^{m-1}$ and the sequence $1, x, x^2, x^3, \cdots, x^{n-2}$. The second column is partially obtained by shifting down the first column. The third column is the result from shifting down the second column.

Sunar and Koç ([12]) made an important observation about the construction of an each row Z_i for $0 \leq i \leq m-1, k \neq n$ of the matrix Z using the above characteristic of the reduction array:

An each row of the product matrix Z can be obtained from Z_n by rewiring.

Sunar and Koç partitioned the product matrix Z into two matrices, that is,

$$Z = X + Y.$$

The matrix X is a $m \times m$ Toeplitz matrix and a lower triangular. On the other hand, the $m \times m$ matrix Y represents the terms obtained through the reduction and is constructed using the reduction array. Algorithm 1 is easily induced by Theorem 1 in [12] and outputs the product matrix Z. The multiplication result c is obtained by the matrix multiplication of Z and b.

Example 1. Let $p(x) = x^5 + x^2 + 1$, $A(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ and $B(x) = b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 \in GF(2^5)$. Then

$$M = \begin{pmatrix} a_0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 \\ a_4 & a_3 & a_2 & a_1 & a_0 \\ 0 & a_4 & a_3 & a_2 & a_1 \\ 0 & 0 & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & a_4 & a_3 \\ 0 & 0 & 0 & 0 & a_4 \end{pmatrix}, \quad X = \begin{pmatrix} a_0 & 0 & 0 & 0 & 0 \\ a_1 & a_0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 & 0 \\ a_4 & a_3 & a_2 & a_1 & a_0 \end{pmatrix}, \quad T = \begin{pmatrix} 0 & a_4 & a_3 & a_2 & a_1 \\ 0 & 0 & a_4 & a_3 & a_2 \\ 0 & 0 & 0 & a_4 & a_3 \\ 0 & 0 & 0 & 0 & a_4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

$$Z = X + Y = \begin{pmatrix} a_0 & a_4 & a_3 & a_2 & a_1 + a_4 \\ a_1 & a_0 & a_4 & a_3 & a_2 \\ a_2 & a_1 + a_4 & a_0 + a_3 & a_4 + a_2 & a_3 + a_1 + a_4 \\ a_3 & a_2 & a_1 + a_4 & a_0 + a_3 & a_4 + a_2 \\ a_4 & a_3 & a_2 & a_1 + a_4 & a_0 + a_3 \end{pmatrix}.$$

We obtain the multiplication of two elements in a finite field by

$$\begin{aligned} c &= Z \cdot b \\ &= \begin{pmatrix} a_0 & a_4 & a_3 & a_2 & a_1 + a_4 \\ a_1 & a_0 & a_4 & a_3 & a_2 \\ a_2 & a_1 + a_4 & a_0 + a_3 & a_4 + a_2 & a_3 + a_1 + a_4 \\ a_3 & a_2 & a_1 + a_4 & a_0 + a_3 & a_4 + a_2 \\ a_4 & a_3 & a_2 & a_1 + a_4 & a_0 + a_3 \end{pmatrix} \cdot \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \\ &= \begin{pmatrix} a_0 \cdot b_0 & + & a_4 \cdot b_1 & + & a_3 \cdot b_2 & + & a_2 \cdot b_3 & + & (a_1 + a_4) \cdot b_4 \\ a_1 \cdot b_0 & + & a_0 \cdot b_1 & + & a_4 \cdot b_2 & + & a_3 \cdot b_3 & + & a_2 \cdot b_4 \\ a_2 \cdot b_0 & + & (a_1 + a_4) \cdot b_1 & + & (a_0 + a_3) \cdot b_2 & + & (a_4 + a_2) \cdot b_3 & + & (a_3 + a_1 + a_4) \cdot b_4 \\ a_3 \cdot b_0 & + & a_2 \cdot b_1 & + & (a_1 + a_4) \cdot b_2 & + & (a_0 + a_3) \cdot b_3 & + & (a_4 + a_2) \cdot b_4 \\ a_4 \cdot b_0 & + & a_3 \cdot b_1 & + & a_2 \cdot b_2 & + & (a_1 + a_4) \cdot b_3 & + & (a_0 + a_3) \cdot b_4 \end{pmatrix}. \end{aligned}$$

3 A New Bit-Parallel Architecture

Sunar and Koç ([12]) showed that the Mastrovito type multiplier using a general trinomial of the form $x^m + x^n + 1$ requires $m^2 - 1$ XOR and m^2 AND gates when $n \neq m/2$. That architecture has the time complexity, $T_A + (\lfloor (m-2)/(m-n) \rfloor + 1 + \lceil \log_2 m \rceil) T_X$.

We propose an efficient multiplication method as a point of view of the time complexity. In order to reduce the total time delay of the circuit, we decompose matrix Z into three matrices S, R, Q and multiply three matrices by b in parallel.

In [12], $Z = X + Y = X + T + T1 + U + U1$. Each entry of upper partial rows of $T + T1$ and lower partial rows of $T + U + X$ has one addition, because that $T1 = T[\uparrow (m-n)]$ and $U = T[\downarrow n]$. Each entry of lower partial rows of $T + U + U1$ has two additions, since $U1 = U[\rightarrow (m-n)]$. Every entry of lower partial rows of $T + U + U1$ contains the entry of upper partial rows of $T + T1$. In *Example*1, the entry of 2nd row and 4th column is $a_3 + a_1 + a_4$ and the entry

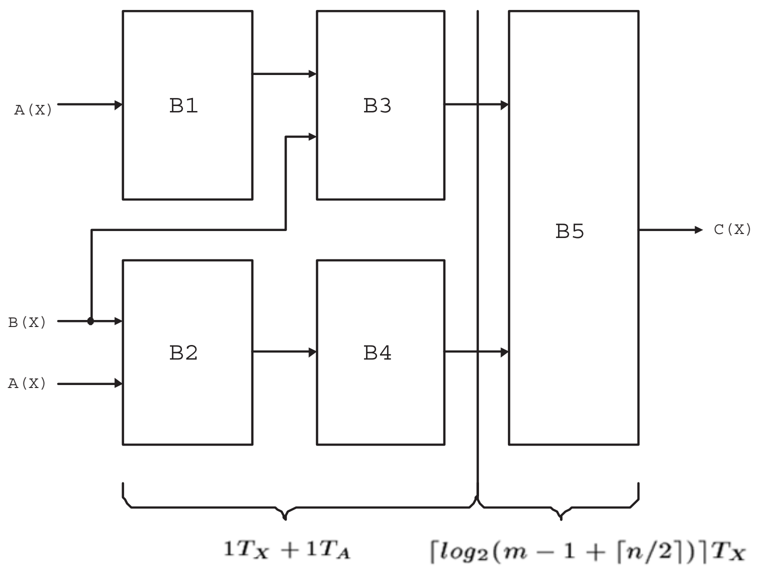


Fig. 1. Bit Parallel multiplier architecture

of 0th row and 4th column is $a_1 + a_4$. In that case, we separate these entries into a single element and an entry of upper partial rows of $T + T1$.

The matrix Q is generated from the characteristic of $T + T1$ and $T + U + X$, whose entries have one addition. The entries of R are obtained from the characteristic of $T + U + U1$ excluding a single element in entries having two addition. The matrix S consists of a single entry. Z is decomposed by $Z = S + Q + R$. $Z \cdot b$ can be computed by $Z \cdot b = S \cdot b + (Q + R) \cdot b$. In that case, we calculate $S \cdot b$ and construct the matrix Q in parallel.

The Bit Parallel structure.
The parallel realization of the proposed method is organized as the following Figure 1.

(1) **The Process of $B1$ and $B3$**
The process computes a combinational logic of $(Q + R) \cdot b$. Entries of the matrix of Q iteratively appeared, since the non-zero part of Q and R is generated by Q_n. That reason is similar to generating Z from Z_n. Q_n is obtained by $X_n + T_n + U_n$ excluding a single entry and some entries that will have two additions.

(2) **The Process of $B2$ and $B4$**
The matrix S has single entries. $B2$ Block computes a bit-wise multiplication of an entry of S and b. In order to synchronize $B1$, $B3$ Block with $B2$, $B4$ Block in the bit parallel structure, $B4$ Block adds some output of $B2$ Block once.

(3) **The Process of** $B5$
In $B5$ Block, the remaining additions are executed using the tree structure of XOR.

We use *Example* 1 to illustrate the proposed architecture. The matrix Q, S and R are

$$Q = \begin{pmatrix} 0 & 0 & 0 & 0 & a_1 + a_4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & a_1 + a_4 & a_0 + a_3 & a_4 + a_2 & 0 \\ 0 & 0 & a_1 + a_4 & a_0 + a_3 & a_4 + a_2 \\ 0 & 0 & 0 & a_1 + a_4 & a_0 + a_3 \end{pmatrix},$$

$$S = \begin{pmatrix} a_0 & a_4 & a_3 & a_2 & 0 \\ a_1 & a_0 & a_4 & a_3 & a_2 \\ a_2 & 0 & 0 & 0 & a_3 \\ a_3 & a_2 & 0 & 0 & 0 \\ a_4 & a_3 & a_2 & 0 & 0 \end{pmatrix},$$

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_1 + a_4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

We define the coordinated notations in order to simply express the proposed multiplier architecture(Figure 3). Output of $B1$ Block makes the matrix Q_2 and the nonzero entries of Q_2 are [1,4], [0,3], [4,2], where [i,j] means $a_i + a_j$. $B2$ Block generates the product S_k and b i.e. $a_i \cdot b_j$, in terms of $(i,\ j)$ as follows:

$$\begin{cases} (0,0),\ (4,1),\ (3,2),\ (2,3) & \text{if } k = 0, \\ (1,0),\ (0,1),\ (4,2),\ (3,3),\ (2,4) & \text{if } k = 1, \\ (2,0),\ (3,4) & \text{if } k = 2, \\ (3,0),\ (2,1) & \text{if } k = 3, \\ (4,0),\ (3,1),\ (2,2) & \text{if } k = 4. \end{cases}$$

$B3$ Block generates multiplication of output of $B1$ Block and partial coefficients of $B(x)$. If $(a_i + a_j) \cdot b_k$ is arranged in terms of $< i,\ j,\ k >$, the output of $B3$ Block is

$$\begin{aligned} &< 1,4,1 >,\ \ < 1,4,2 >,\ \ < 1,4,3 >,\ \ < 1,4,4 >, \\ &< 0,3,2 >,\ \ < 0,3,3 >,\ \ < 0,3,4 >, \\ &< 4,2,3 >,\ \ < 4,2,4 > . \end{aligned}$$

$B4$ Block processes an addition of each kth row of $S \cdot b$ once in parallel.

$$\begin{cases} (0,0) + (4,1),\ (3,2) + (2,3) & \text{if } k = 0, \\ (1,0) + (0,1),\ (4,2) + (3,3),\ (2,4) & \text{if } k = 1, \\ (2,0) + (3,4) & \text{if } k = 2, \\ (3,0) + (2,1) & \text{if } k = 3, \\ (4,0) + (3,1),\ (2,2) & \text{if } k = 4. \end{cases}$$

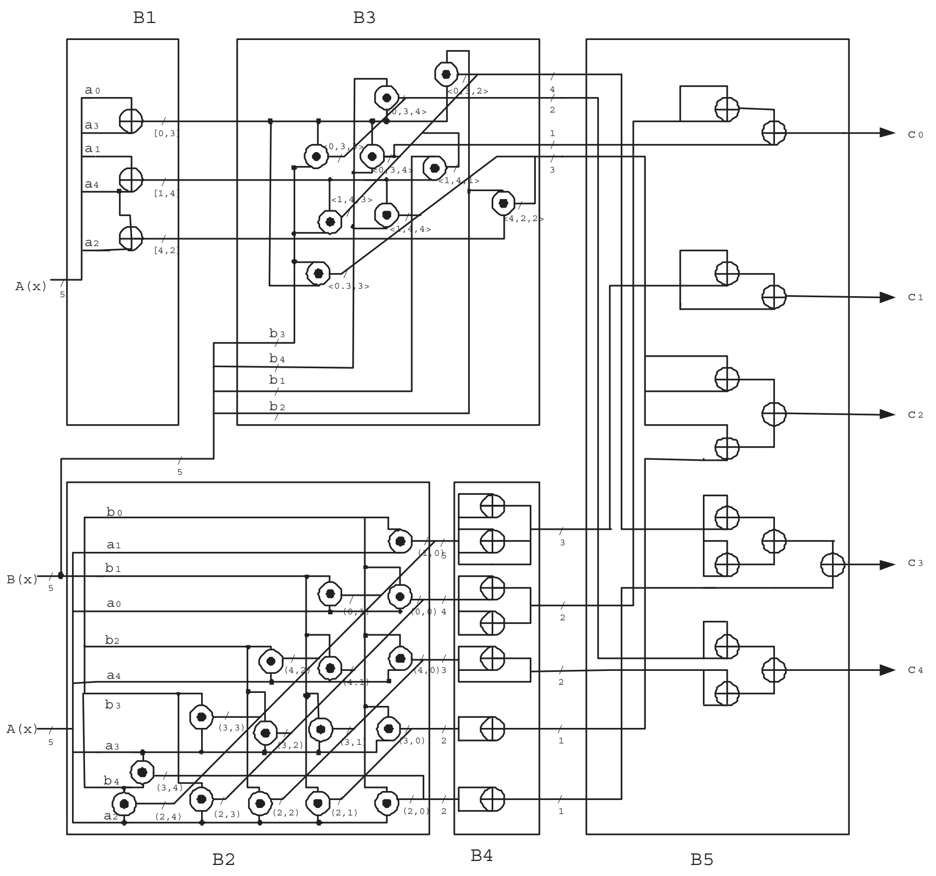


Fig. 2. Bit Parallel multiplier for $p(x) = x^5 + x^2 + 1$

$B5$ Block outputs $A(x) \times B(x)$ after conducting a sum of $B4$ Block and $B3$ Block by using XOR tree structures. The proposed bit-parallel multiplier for this example is organized as the following Figure 3.

4 Complexity Analysis of Proposed Multiplier

In order to analyze the total space and the time complexity, we will determine the complexity for each Block. $B1$ Block and $B4$ Block need XOR gates, $B2$ Block and $B3$ Block need AND gates. Looking into each Block, $B1$ Block is determined by Q_n and Q_n is obtained by $X_n + T_n + U_n$. We denote $|\cdot|$ by the number of nonzero entries. Since X_n and T_n have no intersection, $|Q_n| = |U_n| - |U1_n| = m - 1 - (n - 1) = m - n$. It can be seen that implementation of $B1$ Block requires $m - n$ XOR gates.

The output of $B1$ Block and the coefficients of $B(x)$ are multiplied in $B3$ Block. The number of entries of Q is counted by the number of entries of U and $T1$ except the number of entries of $U1$. Since $T1 = T[\uparrow (m-n)]$ and $U1 = U[\rightarrow (m-n)] = (T[\downarrow n])[\rightarrow (m-n)]$, $T1$ and $U1$ have $n-1(= m-1-(m-n))$ nonzero rows. Thus $|T1| = |U1|$ and

$$\begin{aligned} |Q| &= |U| + |T1| - |U1| = |U| \\ &= (m-1+n)(m-n)/2. \end{aligned}$$

$B2$ Block is determined by the number of entries of S. Since $S = Z - Q - R$, $|R|$ means the number of single elements of entry having two addition of Z and $|Q| = |U|$,

$$\begin{aligned} |S| &= |Z - Q - R| = |Z| - (|Q| + |R|) + |R| \\ &= |Z| - |U| = m^2 - (m-1+n)(m-n)/2. \end{aligned}$$

The space complexity of $B4$ Block depends on the number of entries of row of S. Since the upper n rows of S is constructed by $Z_i - Q_i$, $0 \le i \le n-1$, the number of entries of S_i is $m+1-(n-i)$. Since the middle n rows of S is obtained by $Z_i - Q_i - R_i$, $n \le i \le 2n-1$, the number of entries of S_i is n. Since the lower $m-2n$ rows of S is generated by $Z_i - Q_i$, $2n \le i \le m-1$,the number of entries of S_i is $n+i$. The space complexity of $B4$ Block is

$$\lfloor n/2 \rfloor \cdot n + \sum_{i=1}^{n} \lfloor (m-n-i)/2 \rfloor + \sum_{i=1}^{m-2n} \lfloor (n+i)/2 \rfloor .$$

$B5$ Block is determined by an addition between outputs of $B3$ Block and outputs of $B4$ Block. It can be seen that implementation of $B3$ Block is needed XOR gates as follows:

$$\begin{aligned} &\sum_{i=1}^{n} (\lceil (m-n+i)/2 \rceil + \lceil n/2 \rceil) \\ &+ m(m-1)/2 - m + \sum_{i=1}^{m-2n} (\lceil (n+i)/2 \rceil). \end{aligned}$$

The time complexity of $B1$ Block and $B4$ Block is $1T_X$ and the time complexity of $B2$ Block and $B3$ Block is $1T_A$. The time complexity depends on the longest path between outputs of $B3$ Block and outputs of $B4$ Block. Since the longest path of Z is Z_n, $B5$ Block requires the time complexity as follows:

$$\lceil \log_2(m-1+\lceil n/2 \rceil) \rceil \, T_X .$$

5 Comparison and Discussion

Table 2 contains a numerical complexity formula of comparison between the proposed multiplier and the existing multipliers in [12] and [14]. In [14], Tong Zhang and K. Parhi state lower XOR complexity with the delay increasing. We quote notations from [14]. The notations are

Table 2. The complexity comparison of parallel multipliers with $p(x) = x^m + x^n + 1$

		The existing multiplier ([12])
The time Complexity		$T_A + (\lfloor (m-2)/(m-n) \rfloor + 1 + \lceil \log_2(m) \rceil) T_X$
The space Complexity	AND gate	m^2
	XOR gate	$m^2 - 1$
		The existing multiplier by using hybrid tree ([14])
The time Complexity		$T_A + (t_h + 2 + \lceil \log_2(m) \rceil) T_X$
The space Complexity	AND gate	m^2
	XOR gate	$(m + t_h + h - 1)(m-1) + (\sum_{i=2}^{h} w_i - 2^{t_h} + 2)(m-n)$
		The proposed multiplier
The time Complexity		$T_A + (1 + \lceil \log_2(m - 1 + \lceil n/2 \rceil) \rceil)\ T_X$
The space Complexity	AND gate	m^2
	XOR gate	$m^2 + (n^2 - 3n)/2$

- $t_h = \lfloor (k+1) \rfloor$, where $k = log_2 \lfloor (m-2)/(m-n) \rfloor$.
- h is the Hamming weight of $k+1$.
- $w_i = \sum_{i=j}^{h} 2^{t_i}, 1 \le j \le h$.
- $k + 1 = 2^{t_h} + 2^{t_{h-1}} + \cdots + 2^{t_1}$, where $t_h > t_{h-1} > \cdots > t_1$.

The proposed multiplier is more efficient of the time complexity than the existing multipliers ([12, 14]). The number of AND gate of the existing multipliers and the proposed multiplier is equal. We compare the space complexity with only XOR gates. Table 3 shows the ratio of an improvement measure between

Table 3. Comparing a ratio of complexity in parallel multipliers

Degree	Degree	The Space Complexity			The Time Complexiy		
m	n	Existing ([12])	Proposed	*Ratio*^{1}	Existing([12])	Proposed	*Ratio*^{2}
113	9	12,768	12,796	0.21%	9	8	11.11%
132	17	17,423	17,543	0.68%	10	9	10.00%
135	11	18,224	18,269	0.24%	10	9	10.00%
167	6	27,888	27,898	0.05%	10	9	10.00%
170	11	28,899	28,944	0.12%	10	9	10.00%
191	9	36,480	36,508	0.07%	10	9	10.00%
193	15	37,248	37,339	0.24%	10	9	10.00%
233	74	54,288	56,916	4.83%	10	10	0%
239	36	57,120	57,715	1.04%	10	9	10.00%
289	21	83,520	83,710	0.22%	11	10	9.09%
359	68	128,880	131,091	1.71%	11	10	9.09%
409	87	167,280	170,935	2.18%	11	10	9.09%
431	120	185,760	192,781	3.77 %	11	10	9.09 %

the existing multiplier and the proposed multiplier.

$$\text{Ratio}^1 = \frac{\text{Proposed complexity of XOR - Existing complexity of XOR}}{\text{Existing complexity of XOR}},$$

$$\text{Ratio}^2 = \frac{\text{Existing time complexity - Proposed time complexity}}{\text{Existing time complexity}}.$$

Table 3 shows that the proposed multiplier has lower the time complexity than that of the existing multiplier([12]) because an efficiency of the time complexity is increased to 9.036% on an average but an efficiency of the space complexity is reduced to 1.18% on an average. Our proposed multiplier is more suitable for applications requiring higher speed such as SSL accelerator, VPN, etc.

References

[1] G. H. Golub and C. F. van Loan, Matrix Computations, The Johns Hopkins University Press, 1996.

[2] A. Halbutoğullari and Ç. K. Koç, "Mastrovito multiplier for general irreducible polynomials", Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, Lecture Notes in Computer Science No. 1719, pages 498-507, 1999. 128

[3] IEEE P1363. Standard Specifications for Public-Key Cryptography.Institute of Electrical and Electionics Engineers, 2000. 128

[4] C. K. Kǫc and B. Sunar, "Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields", IEEE Transactions on Computers, 47(3):353-356, March 1998.

[5] R. Lidl and H. Niederreiter, Introduction to Finite Fields and Their Applications, New York, NY: Cambridge University Press, 1994. 127

[6] E. D. Mastrovito, "VLSI architectures for Computation in Galois Fields", PhD thesis, Linkoping University, Department of Electrical Engineering, Linkoping, Sweden, 1991. 128

[7] E. D. Mastrovito, "VLSI architectures for multiplication over finite field". In T.Mora, editor, Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, 6th International Conference, AAECC-6, Lecture Notes in Computer Science, No.357, pages 297-309, Rome, Italy, July 1998. 128

[8] A. J. Menezes, editor, Applications of Finite Fields, Boston, MA: Kluwer Academic Publishers, 1993. 127

[9] A. J. Menezes, Elliptic Curve Public Key Cryptosystems, Boston, MA: Kluwer Academic Publishers, 1993. 127

[10] C. Parr, "A New architecture for a parallel finite field multiplier with low complexity based on composite fields", IEEE Transactions on Computers, 45(7):856-861, July 1996.

[11] SEC2, Recommended Elliptic Curve Domain Parameters, September 20, 2000.

[12] B. Sunar and C. K. Kǫc, "Mastrovito Multiplier for All Trinomial", IEEE Transactions on Computers, 48(5):522-527, May 1999. 127, 128, 129, 130, 131, 132, 136, 137, 138

[13] ANSI X.9.63-1998, Public Key Cryptography for the Financial Services Industry: Elliptic Curve Digital Signature Algorithm (ECDSA).

[14] Tong Zhang and Keshab K. Parhi, "Systematic Design of Original and Modified Mastrovito Multiplier for General Irreducible Polynomials", IEEE Transactions on Computers, 50(7): 734 -749, July 2001 . 128, 136, 137

Architecture for an Elliptic Curve Scalar Multiplication Resistant to Some Side-Channel Attacks

Joong Chul Yoon[1], Seok Won Jung[2], and Sungwoo Lee[1]

[1] C&M, SystemLSI Division, Samsung Electronics Co. Ltd.
{joongchul.yoon,crypto.lee}@samsung.com
[2] Center for Information Security Technologies(CIST), Korea University
jsw@cist.korea.ac.kr

Abstract. This paper describes a design of an elliptic curve scalar multiplication and finite field arithmetic.
The scalar multiplication design resists to Simple Power Analysis(SPA) and solves performance problem induced by SPA countermeasure. Izu and Takagi[9] proposed a parallel multiplication method resistant against SPA. When it is implemented in parallel with two processors, the computing time for n-bit scalar multiplication is $1\cdot$(point doubling) + $(n-1)\cdot$(point addition). Although our design uses one multiplier and one inverter for finite field operation, it takes $2n\cdot$(inversion) to compute n-bit scalar multiplication. If our algorithm utilizes two processors, the computation time is n(point addition) which is almost same as Izu and Takagi's result.
The proposed inverter is resistant to Timing Analysis(TA) and Differential Power Analysis(DPA). López and Dahab[13] argued that for $GF(2^n)$, projective coordinates perform better than the affine coordinates do when inversion operation is more than 7 times slower than the multiplication operation. Speed ratio of the proposed inverter to the proposed multiplier is 6. Thus, the proposed architecture is efficient on the affine coordinates.

Keywords. Finite field, Elliptic curve cryptosystems, Side-channel attack, Parallel Architecture, Montgomery inverse algorithm.

1 Introduction

A scalar multiplication is a basic building block for Elliptic Curve Cryptosystems(ECC) like ECDH and ECDSA. Lots of research have been done about the efficiency for the scalar multiplication.

After Kocher has introduced Timing Attack[11], various low-cost side channel attacks have been proposed[1][12].

Side-channel attack using power consumption of a cryptographic device, e.g. smart card is divided into two methods, Simple Power Analysis(SPA) and Differential Power Analysis(DPA). Kocher, et al[12] firstly had presented the attacks to a symmetric cipher, DES. These methods are easily extended to non-symmetric cryptosystems like RSA, ECC, etc.

Recently ECC has received attention from cryptographic community more than RSA. To prevent DPA, various randomization methods are proposed; scalar multiple randomization[2][5][17], blinding the input point[2] and a randomized projective coordinate[2]. But some of these countermeasures are broken[15] or vulnerable to SPA attack[16][5].

2 Motivations and Results

2.1 Motivations

Coron[2] first introduced a SPA-preventing method which uses the dummy operation in a scalar multiplication. Although it prevented SPA effectively, it had performance problem; if n is the bit-length of a scalar multiple, then Coron's algorithm takes $(n-1)$(point addition)+ $(n-1)$(point doubling). Joye and Quisquater[10] proposed another SPA-countermeasure using a single formula for point addition and doubling by using special representations of certain elliptic curves over prime fields. Their approach also induces some performance penalty.

To resolve the performance problem induced by SPA countermeasures, several methods were introduced. Hasan[7] proposed an efficient SPA-resistant scalar multiplication algorithm. It was limited to only Koblitz Curves. In PKC02, Izu and Takagi[9] showed a scalar multiplication method which can be parallelized under the computing environment using two processors for point addition and point doubling, respectively. Fisher et al[3] proposed parallel multiplication method using two finite field multipliers.

Due to these facts, we have the following motivations for developing a SPA-resistant design.

– It resolves the performance problem.

Okeya and Sakurai[15] used the Montgomery-form elliptic curve where addition and doubling don't depend on the specific bit value of the scalar multiple. Although the computation time per bit of the scalar multiple is $10.2M$, where M is one multiplication time on a finite field, curves suitable for their method should have group order divisible by 4 and be defined over only $GF(p)\ \ (p \geq 5)$. Thus, their method is not applicable to the NIST and SECG recommended curves given in [14][18], which are encouraged to use in order to achieve interoperability.

– It is applicable to the NIST[14] and SECG[18] recommended curves.

The above scalar multiplication algorithms[9][3] used projective coordinates. To convert the representation from projective coordinates to affine coordinates, inversion computation of finite field is required. The ratio of inversion time to multiplication time is important in the decision whether affine or projective coordinates are to be employed. López and Dahab[13] argued that for $GF(2^n)$, projective coordinates perform better than affine coordinates do when inversion

Table 1. The computing time of the proposed design

size of finite field ($GF(2^n)$)	clock (cycles)	timing (at 3Mhz)	timing (at 5Mhz)	timing (at 10Mhz)
163	318,828	106.2 ms	63.7 ms	31.8 ms
233	651,468	217.1 ms	130.2 ms	65.1 ms

in $GF(2^n)$ is more than 7 times slower than multiplication. The efficiency of inversion depends heavily on its algorithm and implementation. Some software implementation[**?**] showed that ratio of inversion time to multiplication time is 4. By the way, other implementation[6] is 10. Normally, software implementation of inversion is 7 to 10 times slower than multiplication.

Wolkerstorfer[21] proposed an unified architecture which can compute multiplication and inversion. In this architecture, the clock-cycle ratio of inversion to multiplication is about 70. But, in [4][19], they proposed inverter architectures which are 6 or 7 times slower than a multiplier in [20].

In addition to resistance to some side-channel attacks,

– Our design uses a multiplier and an inverter for finite field arithmetic.
– Only one multiplier and only one inverter are used.
– The cost-ratio of inversion to multiplication should be less than 7.

Finally, side-channel attacks are efficiently applied to low-speed devices like smart cards and these devices also have area restriction.

– Our design should be applicable to area-restricted device, so, our design should be as small as possible.

2.2 Results

First, we propose a SPA-resistant scalar multiplication which solves performance loss induced by SPA countermeasure. Actually, our design needs $2n$(inversion) to compute n-bit scalar. Since it is constructed based on the standard formula, there is not any restrictions on elliptic curves defined over $GF(2^n)$.

Second, we construct a multiplier and an inverter for $GF(2^n)$ which resistant to TA and DPA. The proposed multiplier needs exactly n clock cycles to compute $a(x) \times b(x) \mod p(x)$ where $a(x), b(x), p(x) \in GF(2^n)$ and the proposed inverter consumes at most $6n$ clock cycles. Since the speed ratio of the inverter to the multiplier is about 6, the proposed scalar multiplication over affine coordinates is efficient.

Third, our design is applicable to small device. Its implementation result on Virtex E2000 FG1156 of Xilinx is about 60,000 gates which are reasonable for smart card. The computing time of the proposed scalar multiplication is estimated in Table 1. In this table, we assume that the inverter takes exactly $6n$ cycles. Then it takes $12n^2$ cycles for a scalar multiplication.

3 Our Proposed Scalar Multiplication Algorithm

3.1 Finite Field Arithmetic for Point Addition and Point Doubling

The formula for point addition and point doubling can be found in various texts and papers. In this paper, The formula in the standard[8] is adopted

$$E : y^2 + xy = x^3 + ax^2 + b \text{ over } GF(2^n),$$

$$P_0 = (x_0, y_0), P_1 = (x_1, y_1),$$

$$\lambda = \begin{cases} \frac{y_0+y_1}{x_0+x_1}, & \text{if } P_0 \neq P_1, \\ x_1 + \frac{y_1}{x_1}, & \text{if } P_0 = P_1, \end{cases}$$

$$x_2 = a + \lambda^2 + \lambda + x_0 + x_1,$$
$$y_2 = (x_1 + x_2)\lambda + x_2 + y_1.$$

The formulae above give the required arithmetic of finite fields in Table 2. INV denotes the inversion, MUL the multiplication and SQR the squaring in finite fields.

Note : There are other arithmetic of finite fields in elliptic curve operations, e.g., field addition and field subtraction. Since the computation time and implementation cost of these operations are very low, Table 2 does not include field addition and subtraction.

These field arithmetic are used at some order for computing point addition and point doubling as depicted in Figure 1. Since XOR is a low cost operation and has no influence to our analysis that will be explained, each step which is denoted as I, S and M, contains XOR operation for simple description. For example, I denotes inversion computation step for x_1 or $x_0 + x_1$. S and M are squaring and multiplication with XOR operations, respectively.

3.2 Scalar Multiplication Design

The proposed algorithm(Algorithm 1) is modified from a right-to-left double-and-add method(Algorithm 2). At each iteration, point addition(step 3.3 in Algorithm 1) is performed. Thus, this algorithm is resistant against SPA.

Table 2. The required arithmetic of finite fields for point addition and point doubling

	$GF(2^n)$
point addition	$1INV + 2MUL + 1SQR$
point doubling	$1INV + 2MUL + 1SQR$

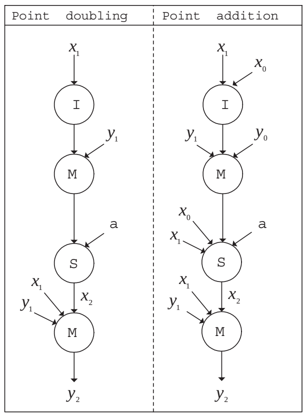


Fig. 1. Order of field arithmetic in point addition and point doubling in $GF(2^n)$

Algorithm 1 :
The proposed algorithm

```
1.   INPUT $P$
2.   $Q[0] \leftarrow O,\ Q[1] \leftarrow O,\ R[0] \leftarrow O,$
     $R[1] \leftarrow P$
3.   For $i$ from 0 to $n-1$
3.1      $R[0] \leftarrow R[1]$
3.2      $R[1] \leftarrow 2R[1]$
3.3      $Q[1] \leftarrow Q[0] + R[0]$
3.4      $Q[0] \leftarrow Q[d_i]$
4.   OUTPUT $Q[0]$
```

Algorithm 2 :
RL method with SPA countermeasure

```
1.   INPUT $P$
2.   $Q[0] \leftarrow P,\ Q[1] \leftarrow O,\ Q[2] \leftarrow O$
3.   For $i$ from 0 to $n-1$
3.1      $Q[2] \leftarrow Q[1] + Q[0]$
3.2      $Q[0] \leftarrow 2Q[0]$
3.3      $Q[1] \leftarrow Q[1 + d_i]$
4.   OUTPUT $Q[1]$
```

In this subsection, two algorithm are compared from the efficiency point of view on hardware implementation. In Algorithm 1, point addition step(step 3.3) can start inversion computation immediately after point doubling operation(step 3.2) finishes the use of inverter. This fact is same as that from Algorithm 2; in Algorithm 2, point doubling step(step 3.2) can start inversion immediately after point addition(step 3.1) finishes its own inversion.

In Algorithm 2, before starting point addition in an iteration, point doubling in previous iteration should be finished since new $Q[0]_x$(where $Q[0] = (Q[0]_x, Q[0]_y)$)is used at inversion computation in point addition. Thus, point addition can't start inversion immediately after finishing inversion in point doubling. Figure 2 explains the above fact about Algorithm 2. As opposed to Algorithm 2, two elliptic operations in Algorithm 1 use different variables; point doubling only uses $R[1]$ and point addition uses $Q[0]$ and $R[0]$. This fact, as

Iter-ation	cycle	Addition	*Q[2]*	Doubling	*Q[0]*
i=0	1*c*	**I**			
	2*c*				
	3*c*				
	4*c*				
	5*c*				
	6*c*				
	7*c*	**M**		**I**	
	8*c*	**S**	New Q[2]x of Q[2]		
	9*c*	**M**	New Q[2]y of Q[2]		
	10*c*				
	11*c*				
	12*c*				
	13*c*			**M**	
	14*c*			**S**	New Q[0]x of Q[0]
i=1	15*c*	**I**		**M**	New Q[0]y of Q[0]
	16*c*				
	17*c*				
	18*c*				
	19*c*				
	20*c*				
	21*c*	**M**		**I**	
	22*c*	**S**	New Q[2]x of Q[2]		
	23*c*	**M**	New Q[2]y of Q[2]		
	20*c*				
	21*c*				
	22*c*				

Fig. 2. The computation procedure of Algorithm 2

described in Figure 3[1], makes point doubling start the inversion computation immediately after point addition finishes the inversion.

Thus, with one multiplier and one inverter, normal SPA-resistant algorithm (Algorithm 2) saves only four-multiplication time for each iteration, but Algorithm 1 saves 6 multiplications as described in Figure 3. Thus, there are only two-inversion time in each iteration in Algorithm 1.

There is a difference between Algorithm 1 and Figure 3 at data transfer step from $R[1]$ to $R[0]$. In implementation aspect, only point addition needs new $R[0]$, so that it does not matter if $R[0]$ gets new value from $R[1]$ before point addition starts the inversion computation.

4 Inverter and Multiplier Design

In this section, we propose a multiplier and an inverter resistant to TA and DPA.

4.1 Multiplier Design

We adopt the following Algorithm 3 for finite field multiplication.

[1] In this subsection, we assume that the ratio of inversion to multiplication is 6. At section 4, detailed description about inverter and multiplier will be provided.

Iter-ation	cycle	Doubling	*R[1]*	Addition	*Q[1]*	*R[0]*
i=1	1*c*	**I**				
	2*c*					
	3*c*					
	4*c*					
	5*c*					
	6*c*					R[0] R[1] (R[0]=P)
	7*c*	**M**		**I**		
	8*c*	**S**	New R[1]x of R[1]			
	9*c*	**M**	New R[1]y of R[1]			
	10*c*					
	11*c*					
	12*c*					
i=2	13*c*	**I**		**M**		
	14*c*			**S**	New Q[1]x of Q[1]	
	15*c*			**M**	New Q[1]y of Q[1]	
	16*c*					
	17*c*					
	18*c*					R[0] R[1] (R[0]=2P)
	19*c*	**M**		**I**		
	20*c*	**S**	New R[1]x of R[1]			
	21*c*	**M**	New R[1]y of R[1]			
	22*c*					

Fig. 3. The computation procedure of the proposed algorithm

Although Algorithm 3 is an ordinary shift-and-add method, the computing time of the architecture does not depend on the number of zeros in operand or the degree of operand. It takes more time to compute $a(x) \cdot b(x) \mod p(x)$ than other efficient algorithms or architectures do, but it does not leak information which can be used by TA.

As shown in Figure 4 and Algorithm 3, the proposed multiplier finishes a computation after n clock cycles and its size is so small that it is suitable for small device.

```
Algorithm 3 : the multiplication algorithm
1.   INPUT $a(x), b(x), p(x)$ in $GF(2^n)$
2.   $c(x) \leftarrow 0$
3.   For $i$ from $n-1$ to 0
3.1      $c(x) \leftarrow c(x) \cdot x$
3.2      $c(x) \leftarrow c(x) + c_n \cdot p(x) + b_i \cdot a(x)$
4.   OUTPUT $c(x)$
```

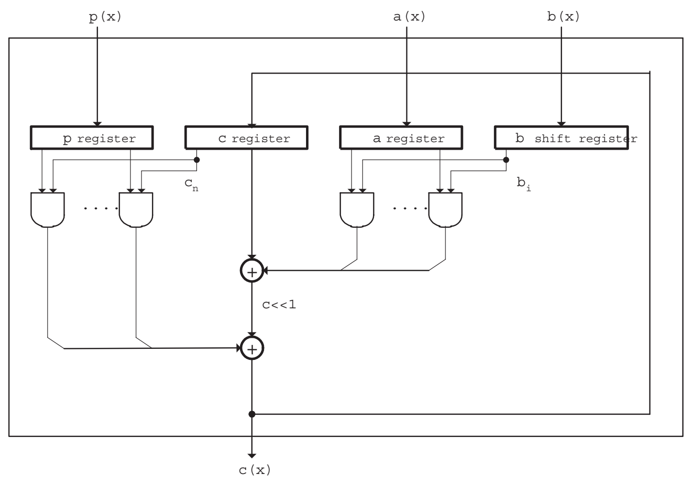


Fig. 4. The architecture of the multiplication algorithm resistant to TA

4.2 Inverter Design

We design an inverter architecture based on the phase I of Montgomery inverse algorithm presented in [19]. That algorithm ends with $a(x)^{-1}x^k$ for $a(x) \in GF(2^n)$ and the k is $\deg(a(x)) \leq k \leq \deg(p(x)) + \deg(a(x)) + 1$, i.e., $k \leq 2n$. Although its computation time per input value is different, computation time is always same if identical value is used repeatedly. So, it is weak to TA. Since the inversion is the main part in point addition and doubling, weakness of inversion to TA also influence the scalar multiplication, so it can be also vulnerable to TA.

To resist to TA, inverter should have variable computation time for fixed input value. Thus, we propose a modified Algorithm 4.

Algorithm 4 adds the redundant operation(step 4.2) which is computed in random number of times. Although $k \leq 2n$ in the original algorithm in [19], Algorithm 4 increases the degree of x^k, i.e., $a(x)^{-1}x^m = a(x)^{-1}x^k x^{m-k}$ for $a(x) \in GF(2^n)$ where $2n < m < 3n$.

Algorithm 4 ends with $a(x)^{-1}x^m$, $2n < m < 3n$ for $a(x) \in GF(2^n)$. $a(x)^{-1}x^m$ can be easily converted to $a(x)^{-1}$ by the following congruent relation;

$$a(x)^* = a(x)^{-1}x^m + c_0 \cdot p(x) \mod p(x) = a(x)^{-1} \cdot x^m,$$

where c_0 is the constant term of $a(x)^{-1}x^m$. Then, the constant term of $a(x)^*$ is zero. Thus we can obtain

$$a(x)^{-1}x^{m-1} = a(x)^{-1}\frac{x^m}{x} = a(x)^*\frac{1}{x}$$

which is implemented as just bit-shift. After m iterations, we have $a(x)^{-1}$. The detail of the algorithm is as follows:

Algorithm 4 : the modified Montgomery inverse algorithm resistant to TA

```
1.      INPUT $a(x), p(x)$ where $\deg(a(x)) \le \deg(p(x))$
2.      $u(x) \leftarrow p(x)$,  $v(x) \leftarrow a(x)$,  $r(x) \leftarrow 0$,  $s(x) \leftarrow 1$
3.      $m \leftarrow$ random value, $2n < m < 3n$
4.      while($k < m$)
4.1       if ($v(x) \ne 0$)
4.1.1       if ($u_0 \ne 0$) $u(x) \leftarrow u(x)/x, s(x) \leftarrow x \cdot s(x)$
4.1.2       else if ($v_0 \ne 0$) $v(x) \leftarrow v(x)/x, r(x) \leftarrow x \cdot r(x)$
4.1.3       else if ( $\deg(u(x)) > \deg(v(x))$ ) then
              $u(x) \leftarrow (u(x) + v(x))/x$
              $r(x) \leftarrow r(x) + s(x)$
              $s(x) \leftarrow x \cdot s(x)$
4.1.4       else $v(x) \leftarrow (v(x) + u(x))/x$
              $s(x) \leftarrow s(x) + r(x)$
              $r(x) \leftarrow x \cdot r(x)$
4.2       else
4.2.1       if ($r_n = 1$) $r(x) \leftarrow r(x) + p(x)$
4.2.2       $r(x) \leftarrow x \cdot r(x)$
4.3       $k \leftarrow k + 1$
5.      if ( $\deg(r(x)) = \deg(p(x))$ ) $r(x) \leftarrow r(x) + p(x)$
6.      OUTPUT $r(x)$
```

The number of steps in Algorithm 5 is different due to m. Thus, Algorithm 4 and 5 can calculate $a(x)^{-1}$ and are resistant to TA.

Now we consider Algorithm 4 from DPA point of view. Before $v(x)$ is zero, Algorithm 4 always goes to step 4.1. So, although some randomness of time is added to Algorithm 4, DPA can be applied using the power consumption curve made during the computation of step 4.2. To prevent DPA weakness, we must insert another randomness to Algorithm 4.

By the random factor, R, Algorithm 6 goes randomly to step 4.2 or step 4.3. By the way, one of our motivation is to achieve the cost-ratio of inversion to multiplication. To achieve this motivation, Algorithm 6 goes to step 4.3 in probability of one fourth. If the probability is one second(=1/2), the maximum number of iteration(value m) can be $4n$ since step 4.2 in Algorithm 4 finishes at $2n$ iterations in some cases. But we should use Algorithm 5 to get $a(x)^{-1}$. Thus the total number of iteration can be $8n$. It is not a desirable result.

Algorithm 5 : Algorithm for computing $a(x)^{-1}$ from $a(x)^{-1}x^m$

```
1.   INPUT $a(x)^{-1}x^m$
2.   $c(x) \leftarrow a(x)^{-1}x^m$
3.   For $i$ from $m-1$ to 0
3.1      $c(x) \leftarrow (c(x) + c_0 p(x))/x$
4.   OUTPUT $c(x)$
```

Algorithm 6 : the modified Montgomery inverse algorithm resistant to TA and DPA

```
1.      INPUT $a(x), p(x)$ where $\deg(a(x)) \leq \deg(p(x))$
2.      $u(x) \leftarrow p(x)$,   $v(x) \leftarrow a(x)$,   $r(x) \leftarrow 0$,   $s(x) \leftarrow 1$
3.      $m \leftarrow$ random value, $2n < m < 3n$
4.      while($k < m$ or $v(x) \neq 0$)
4.1       $R \leftarrow$ random number, $0 \leq R \leq 3$
4.2       if ($v(x) \neq 0$ and $R \neq 0$)
4.2.1       if ($u_0 \neq 0$) $u(x) \leftarrow u(x)/x, s(x) \leftarrow x \cdot s(x)$
4.2.2       else if ($v_0 \neq 0$) $v(x) \leftarrow v(x)/x, r(x) \leftarrow x \cdot r(x)$
4.2.3       else if ( $\deg(u(x)) > \deg(v(x))$ ) then
              $u(x) \leftarrow (u(x) + v(x))/x$
              $r(x) \leftarrow r(x) + s(x)$
              $s(x) \leftarrow x \cdot s(x)$
4.2.4       else $v(x) \leftarrow (v(x) + u(x))/x$
              $s(x) \leftarrow s(x) + r(x)$
              $r(x) \leftarrow x \cdot r(x)$
4.3       else
4.3.1       if ($r_n = 1$) $r(x) \leftarrow r(x) + p(x)$
4.3.2       $r(x) \leftarrow x \cdot r(x)$
4.3.3       if ($s_n = 1$) $s(x) \leftarrow s(x) + p(x)$
4.3.4       $s(x) \leftarrow x \cdot s(x)$
4.4       $k \leftarrow k + 1$
5.      if ( $\deg(r(x)) = \deg(p(x))$ ) $r(x) \leftarrow r(x) + p(x)$
6.      OUTPUT $r(x)$
```

If R gets four value(0,1,2 and 3) uniformly during calculation, then the maximum number of iteration is $2n + 2n/3$. By the way, the condition of step 4 is modified as $k < m$ or $v(x) \neq 0$ since it happens that $v(x)$ is not zero and $k \geq m$, then algorithm finishes without correct output value. Thus, Algorithm 6 ends with random steps within $3n$.

Therefore, Algorithm 6 and 5 can calculate $a(x)^{-1}$ within $6n$ iterations and are resistant to TA and DPA.

Although Algorithm 5 and 6 can be implemented in various way, our design focuses on the iteration to achieve the cost ratio. Step 4.2, 4.3 of Algorithm 6 and step 3 of Algorithm 5 are performed in one step. Figure 5 reflects these considerations.

If the method as described in [19] is used at degree checking step, $\deg(u(x)) > \deg(v(x))$ can affect the performance of Algorithm 6. To check $\deg(u(x)) > \deg(v(x))$ with no affecting the performance or iteration step, we use the following method.

Let $deg(u(x)) = (1, 0, \cdots, 0)$, i.e., $deg(u(x))_n = 1$ and $deg(u(x))_{n-1} = \cdots = deg(u(x))_0 = 0$. When the degree of $u(x)$ decreases, $deg(u(x))$ is right-shifted and added 1 at the most significant bit. For example, $deg(u(x)) = (1, 0, \cdots, 0)$ and "$u(x) \leftarrow u(x)/x$" is happened, $deg(u(x)) = (1, 1, 0, \cdots, 0)$ i.e., $deg(u(x))_n = 1$ and $deg(u(x))_{n-1} = 1, deg(u(x))_{n-2} = \cdots = deg(u(x))_0 = 0$. With these

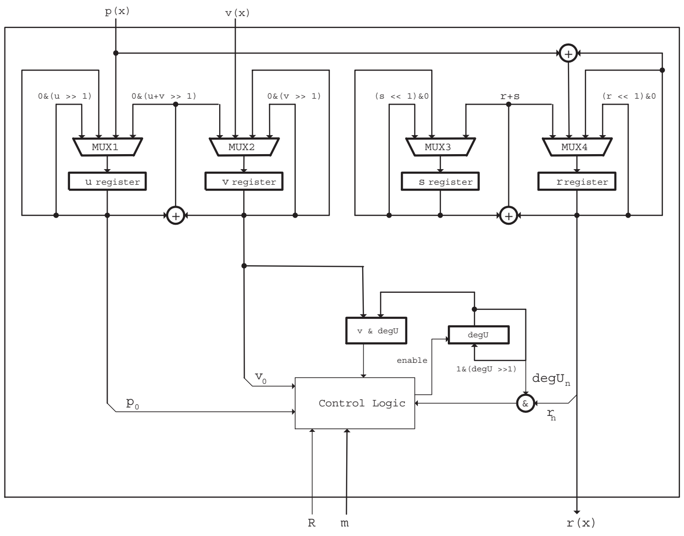


Fig. 5. The architecture of Algorithm 6

value, we can check $\deg(u(x)) \leq \deg(v(x))$ on the fly as follows. If $\deg(u(x)) \leq \deg(v(x))$, then output of bitwise AND of $deg(u(x))$ and $v(x)$ is not zero vector, otherwise it is zero vector. With this method, it is not necessary to check the degree of $v(x)$ before starting inversion and there is no additional clock for degree-checking.

5 Conclusion

In this paper, we presented the scalar multiplication algorithm which can solve the performance loss induced by SPA countermeasure. The inverter which is resistant to TA and DPA also described. The computation time of proposed inverter is 6 times slower than the proposed multiplier. Thus, our inverter and multiplier can be applied to the proposed scalar multiplication over affine coordinates.

Due to the parallel computation of inversion and multiplication, our algorithm increases the randomness for power consumption. DPA would be more difficult than ordinary SPA resistant scalar multiplication.

References

1. D. Boneh, R. DeMillo, and R. Lipton, On the Importance of Checking Cryptographic Protocols for Faults, Advances in Cryptology - EUROCRYPT'97, LNCS1294, pages 37-51, Springer-Verlag, 1997. 139
2. J.-S. Coron, Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems, Workshop on Cryptographic Hardware and Embedded Systems - CHES'99, LNCS1717, pages 292-302, Springer-Verlag, 1999. 140
3. W. Fischer, C. Giraud, E. W. Knudsen and J.-P. Seifert, Parallel scalar multiplication on general elliptic curves over F_p hedged against Non-Differential Side-Channel Attacks, Available from `http://eprint.iacr.org`, 2002. 140
4. A. A.-A. Gutub, A. F. Tenca, E. Savaş, and Ç. K. Koç, Scalable and unified hardware to compute Montgomery inverse in *GF(p)* and $GF(2^n)$, Workshop on Cryptographic Hardware and Embedded Systems - CHES2002, LNCS 2523, pages 484–499, Springer-Verlag, 2002. 141
5. J. C. Ha, and S. J. Moon, Randomized Signed-Scalar Multiplication of ECC to Resist Power Attacks, Workshop on Cryptographic Hardware and Embedded Systems - CHES2002, LNCS 2523, pages 553-565, Springer-Verlag, 2002. 140
6. D. Hankerson, J. Hernandez and A. Menezes, Software Implementation of Elliptic Curve Cryptography Over Binary Fields, Workshop on Cryptographic Hardware and Embedded Systems - CHES2000, LNCS1965, pages 1-24, Springer-Verlag, 2000. 141
7. M. A. Hasan, Power Analysis Attacks and Algorithmic Approaches to their Countermeasures for Koblitz Curve Cryptosystems, Workshop on Cryptographic Hardware and Embedded Systems - CHES2000, LNCS1965, pages 93-108, Springer-Verlag, 2000. 140
8. Institute of Electrical and Electronics Engineers (IEEE), IEEE standard specifications for public-key cryptography, IEEE Std 1363-2000, 2000. 142
9. T. Izu and T. Takagi, A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks, Public Key Cryptography - PKC 2002, LNCS2274, pages 280-296, Springer-Verlag, 2002. 139, 140
10. M. Joye and J. Quisquater, Hessian ellitpic curves and side-channel attacks, Workshop on Cryptographic Hardware and Embedded Systems - CHES2001, LNCS2162, pages 402-410, Springer-Verlag, 2001. 140
11. P. Kocher, Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems, Advances in Cryptology - CRYPTO'96, LNCS1109, pages 104-113, Springer-Verlag, 1996. 139
12. P. Kocher, J. Jaffe, and B. Jun, Differential power analysis, Advances in Cryptology - CRYPTO'99, LNCS1666, pages 388-397, Springer-Verlag, 1999. 139
13. J. López and R. Dahab, Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Precomputation, Workshop on Cryptographic Hardware and Embedded Systems - CHES'99, LNCS1717, pages 316-327, Springer-Verlag, 1999. 139, 140
14. National Institute of Standards and Technology(NIST), Digital Signature Standard(DSS) FIPS PUB 186-2, 2000. 140
15. K. Okeya and K. Sakurai, Power analysis breaks elliptic curve cryptosystems even secure against the timing attack, Progress in Cryptology - Indocrypt 2000, LNCS1977, pages 178-190, Springer-Verlag, 2000. 140
16. K. Okeya and K. Sakurai, On Insecurity of the Side Channel Attack Countermeasure Using Addition-Subtraction Chains under Distinguishability between Addition and Doubling, Information Security and Privacy - ACISP'02, LNCS2384, pages 420-435, Springer-Verlag, 2002. 140

17. E. Oswald and M. Aigner, Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks, Workshop on Cryptographic Hardware and Embedded Systems - CHES2001, LNCS2162, pages 39-50, Springer-Verlag, 2001. 140
18. Certicom Research, Standards for efficient cryptography-SEC2 : Recommended elliptic curve cryptography domain parameters, Available from `http://www.secg.org`, 2000. 140
19. E. Savaş and Ç. K. Koç, Architectures for unified field inversion with applications in elliptic curve cryptography, The 9th IEEE International Conference on Electronics, Circuits and Systems - ICECS2002, IEEE, pages 1155-1158, 2002. 141, 146, 148
20. E. Savaş, A. F. Tenca and Ç. K. Koç, A Scalable and Unified Multiplier Architecture for Finite Fields *GF(p)* and $GF(2^n)$, Workshop on Cryptographic Hardware and Embedded Systems - CHES2000, LNCS1965, pages 281-296, Springer-Verlag, 2000. 141
21. J. Wolkerstorfer, Dual-Field Arithmetic Unit for *GF(p)* and $GF(2^m)$, Workshop on Cryptographic Hardware and Embedded Systems - CHES2002, LNCS 2523, pages 501-515, Springer-Verlag, 2002. 141

Efficient Scalar Multiplication in Hyperelliptic Curves Using A New Frobenius Expansion*

Tae-Jun Park[1], Mun-Kyu Lee[2], and Kunsoo Park[1]

[1] School of Computer Science and Engineering
Seoul National University, Seoul, 151-744, Korea
{tjpark,kpark}@theory.snu.ac.kr
[2] Electronics and Telecommunications Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
mklee@etri.re.kr

Abstract. The Frobenius expansion has been used to speed up scalar multiplication in hyperelliptic curves as it is used in elliptic curves. In this paper we propose a new Frobenius expansion method for hyperelliptic curves that have efficiently computable endomorphisms used in Park, Jeong and Lim [1]. When our method is applied to scalar multiplication for hyperelliptic curves, the number of divisor doublings in a scalar multiplication is reduced to a quarter, while the number of divisor additions is almost the same. Our experiments show that the overall throughputs of scalar multiplications are increased by 15.6–28.3% over the previous algorithm, when the algorithms are implemented over finite fields of odd characteristics.

Keywords. Hyperelliptic Curve, Scalar Multiplication, Frobenius Expansion

1 Introduction

Koblitz [2] suggested the use of Jacobians of hyperelliptic curves for cryptographic applications. Its advantages over the use of elliptic curves are the smaller field sizes and the larger variety of curves that can be used to cryptosystems. The most-time consuming operation in a hyperelliptic curve cryptosystem based on the discrete logarithm problem is a scalar multiplication by an integer k, i.e., computing kD for a divisor D on the Jacobian.

In elliptic curves, Koblitz [3] proposed the curves that are defined over the binary field but whose coordinates are on suitably large extension fields, which are called Koblitz curves. The idea of elliptic Koblitz curves was generalized to hyperelliptic curves by Günter, Lange and Stein [4]. They investigated two special examples of curves of genus 2 defined over the binary field using the Frobenius map. Lange [5] gave a detailed investigation on hyperelliptic Koblitz curves of small genus defined over small fields using the Frobenius map. In Lange [5] and

* This work was supported by the MOST grant M6-0203-00-0039.

Choie and Lee [6] the Frobenius expansion method was generalized to the finite field of any characteristic.

Gallant, Lambert and Vanstone [7] introduced a method using special elliptic curves which have efficiently computable endomorphisms other than Frobenius maps. The idea of their method is to decompose an integer k into two components such that the size of each component is a half of that of k. Park, Jeong and Lim [1] extended this idea to hyperelliptic curves that have efficiently computable endomorphisms.

In this paper we propose a new Frobenius expansion method for hyperelliptic curves with efficiently computable endomorphisms. To compute kD for an integer k and a divisor D, we extend the integer k by the Frobenius endomorphism φ:

$$k = \sum_{i=0}^{n} r_i \varphi^i \ ,$$

where the coefficients r_i are of the form $r_{i0} + r_{i1}\rho + r_{i2}\rho^2 + r_{i3}\rho^3$ or $r_{i0} + r_{i1}\gamma + r_{i2}\gamma^2 + r_{i3}\gamma^3$ ($r_{ij} \in \mathbb{Z}$), and ρ and γ are efficiently computable endomorphisms used in [1]. Park, Lee and Park [8] gave a similar Frobenius expansion method in elliptic curves.

Our method can be used to improve the known scalar multiplication algorithm for hyperelliptic curves that uses the Frobenius expansion [5, 6]. When our method is applied to this algorithm, the number of divisor doublings in a scalar multiplication is reduced to a quarter, while the number of divisor additions is almost the same. Our experiments show that the overall throughputs of scalar multiplications are increased by 15.6–28.3% over the previous algorithm, when the algorithms are implemented over $\mathbb{F}_{p^n}$ where p and n are prime.[1]

2 Preliminaries

2.1 Basic Definitions

We first give the basic definitions about hyperelliptic curves (see [2], [10]). Let $\mathbb{F}_q$ be a finite field of q elements. A hyperelliptic curve of genus g is defined by the equation of the form

$$\mathcal{C} : y^2 + h(x)y = f(x) \ , \tag{1}$$

where $h(x), f(x) \in \mathbb{F}_q[x]$, $\deg_x(h) \leq g$, $f(x)$ monic, $\deg_x(f) = 2g+1$, and there are no solutions $(x, y) \in \bar{\mathbb{F}}_q \times \bar{\mathbb{F}}_q$ which simultaneously satisfy (1) and the partial derivative equations $2y + h(x) = 0$ and $h'(x)y - f'(x) = 0$. Let $\mathbb{K}$ be an extension field of $\mathbb{F}_q$ in $\bar{\mathbb{F}}_q$. The set of $\mathbb{K}$-rational points on $\mathcal{C}$ consists of all points $(x, y) \in \mathbb{K} \times \mathbb{K}$ which satisfy the equation (1), together with a point at

[1] The security implications of the Weil-Descent [9] on these curves are not yet clear. We remark that there is no known attack that significantly reduces the time required to compute hyperelliptic curve discrete logarithms on these curves.

infinity, denoted by ∞. Let $P = (x, y) \neq \infty$ be a point on $\mathcal{C}$. The opposite of P is the point $\tilde{P} = (x, -y - h(x))$.

A divisor is a formal sum $D = \sum_{P \in \mathcal{C}} m_P P$, where $m_P \in \mathbb{Z}$ and $m_p = 0$ for almost all $P \in \mathcal{C}$. The degree of D is the integer $\sum_{P \in \mathcal{C}} m_P$. The set of all divisor, denoted by $\mathbb{D}$, forms an additive group and the set of all divisors of degree 0, denoted by $\mathbb{D}^0$, is a subgroup of $\mathbb{D}$. A divisor $D \in \mathbb{D}^0$ is called a principal divisor if $D = div(R)$ for some rational function $R \in \bar{\mathbb{F}}_q(\mathcal{C})^*$. The set of all principal divisors, denoted by $\mathbb{P}$, is a subgroup of $\mathbb{D}^0$.

The quotient group $\mathbb{J} = \mathbb{D}^0/\mathbb{P}$ is called the Jacobian of the curve $\mathcal{C}$. The Jacobian is an abelian variety whose dimension is the genus of the curve $\mathcal{C}$ (see [11]). By the Riemann-Roch theorem, every divisor $D \in \mathbb{D}^0$ can be uniquely represented as an equivalence class in $\mathbb{J}$ by a reduced divisor of the form $\sum m_i P_i - (\sum m_i)\infty$ with $\sum m_i \leq g$. Due to Mumford [12], a reduced divisor can be represented by a pair of polynomials $u(x)$ and $v(x) \in \mathbb{F}_q[x]$ for which $\deg_x(v) < \deg_x(u) \leq g$, and $v(x)^2 + h(x)v(x) - f(x)$ is divisible by $u(x)$. D is the equivalence class of the GCD of the divisors of the functions $u(x)$ and $v(x) - y$, denoted by $\mathrm{div}(u, v)$.

The addition algorithms in the Jacobian were presented by Koblitz [2], and they are generalization of the earlier algorithms of Cantor [13]. The addition needs $17g^2 + O(g)$ operations in $\mathbb{K}$ and the doubling takes $16g^2 + O(g)$ operations (see [14]). The scalar multiplication by an integer k is denoted by

$$[k]D = \overbrace{D + D + \cdots + D}^{k} \ .$$

The discrete logarithm problem in the Jacobian is the problem that, given two divisors D_1 and D_2, determines an integer $k \in \mathbb{Z}$ such that $D_2 = kD_1$ if such k exists.

2.2 Hyperelliptic Curves with Efficient Endomorphisms

Park, Jeong and Lim [1] collected the following hyperelliptic curves of genus g over $\mathbb{F}_q$ that have efficiently computable endomorphisms.

Example 1. [**5**] Let X be a hyperelliptic curve over $\mathbb{F}_q$ given by (1). The q-th power map, called the Frobenius map, $\varphi : X \longrightarrow X$ denoted by $(x, y) \longrightarrow (x^q, y^q)$ induces an endomorphism on the Jacobian. The characteristic polynomial of the Frobenius map φ is given by

$$P(t) = t^{2g} + a_1 t^{2g-1} + \cdots + a_g t^g + q a_{g-1} t^{g-1} + \cdot + q^{g-1} a_1 t + q^g \ ,$$

where $a_0 = 1$, and $i a_i = S_i a_0 + S_{i-1} a_1 + \cdots + S_1 a_{i-1}$ for $S_i = N_i - (q^i + 1)$, $1 \leq i \leq g$ and $N_i = |X(\mathbb{F}_{q^i})|$.

Example 2. [**15, 16**] Let $p \equiv 1 \pmod 5$ be prime. Consider the hyperelliptic curve X_1 of genus 2 over the field $\mathbb{F}_p$ defined by

$$X_1 : y^2 = x^5 + a \ . \qquad (2)$$

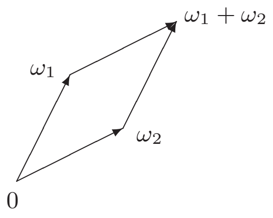


Fig. 1. A 2-dimensional fundamental parallelogram

The morphism ρ defined by $(x, y) \mapsto (\zeta_5 x, y)$ induces an efficient endomorphism on the Jacobian, where ζ_5 is a 5-th root of unity. The characteristic polynomial of ρ is given by $P(t) = t^4 + t^3 + t^2 + t + 1$. The formulae for ρ on the Jacobian are given by

$$\begin{aligned}\rho : [x^2 + a_1 x + a_0, b_1 x + b_0] &\mapsto [x^2 + \zeta_5 a_1 x + \zeta_5^2 a_0\ , \zeta_5^{-1} b_1 x + b_0] \\ [x + a_0, b_0] &\mapsto [x + \zeta_5 a_0, b_0] \\ 0 &\mapsto 0\ .\end{aligned}$$

Example 3. [1] Let $p \equiv 1 \pmod 8$ be prime. Consider the hyperelliptic curve X_2 of genus 2 over the field $\mathbb{F}_p$ defined by

$$X_2 : y^2 = x^5 + a\ x\ . \tag{3}$$

Then, the morphism γ on X_2 defined by $(x, y) \mapsto (\zeta_8^2 x, \zeta_8 y)$ induces an efficient endomorphism, where ζ_8 is a 8-th root of unity. The characteristic polynomial of γ is given by $P(t) = t^4 + 1$. The formulae for γ on the Jacobian are given by

$$\begin{aligned}\gamma : [x^2 + a_1 x + a_0, b_1 x + b_0] &\mapsto [x^2 + \zeta_8^2 a_1 x + \zeta_8^4 a_0\ , \zeta_8^{-1} b_1 x + \zeta_8 b_0] \\ [x + a_0, b_0] &\mapsto [x + \zeta_8^2 a_0, \zeta_8 b_0] \\ 0 &\mapsto 0\ .\end{aligned}$$

We introduce an important property of the endomorphism rings of Jacobians. Let $E = End(X) \otimes \mathbb{Q}$. By J. Tate [17], the characteristic polynomial of the Frobenius map φ has no double roots if and only if $E \cong \mathbb{Q}(\varphi)$ and $[E : \mathbb{Q}] = 2g$.

2.3 Lattices

By a 2 (resp. 4)-dimensional lattice in the complex plane $\mathbb{C}$, we shall mean a subgroup which is free of dimension 2 (resp. 4) over $\mathbb{Z}$. If $\{\omega_1, \omega_2\}$ is a basis of a 2-dimensional lattice L over $\mathbb{Z}$, then we write $L = [\omega_1, \omega_2]$. Similarly, if $\{\omega_1, \omega_2, \omega_3, \omega_4\}$ is a basis of a 4-dimensional lattice L over $\mathbb{Z}$, then we write $L = [\omega_1, \omega_2, \omega_3, \omega_4]$.

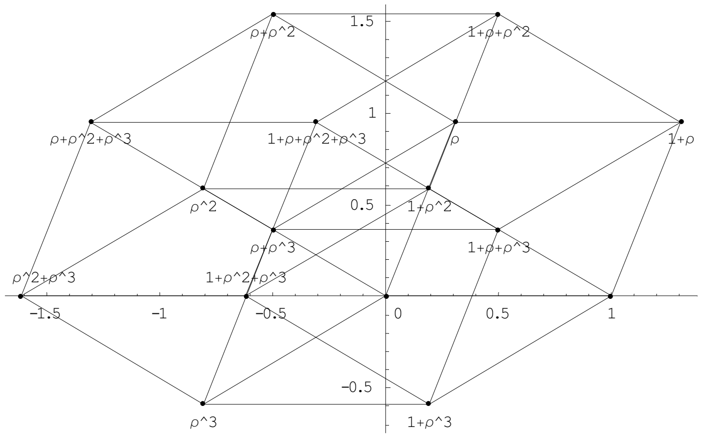


Fig. 2. The fundamental parallelogram of $\mathbb{Z}[\zeta_5]$, where $\rho = \zeta_5$

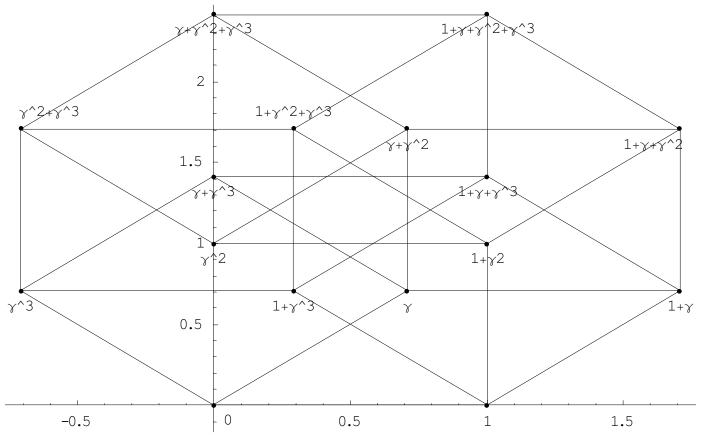


Fig. 3. The fundamental parallelogram of $\mathbb{Z}[\zeta_8]$, where $\gamma = \zeta_8$

The fundamental parallelogram for $L = [\omega_1, \omega_2]$ is the set consisting of all points $t_1\omega_1 + t_2\omega_2$, where $0 \leq t_i \leq 1$ (see Fig. 1). The fundamental parallelogram for $L = [\omega_1, \omega_2, \omega_3, \omega_4]$ is the set consisting of all points $t_1\omega_1 + t_2\omega_2 + t_3\omega_3 + t_4\omega_4$, where $0 \leq t_i \leq 1$ (see Fig. 2, 3).

For example, $\mathbb{Z}[i] = \{a + bi \mid a, b \in \mathbb{Z}\}$ is the ring of all algebraic integers of $\mathbb{Q}(i)$. We can consider $\mathbb{Z}[i]$ as the lattice $L = [1,\ i]$. By [18], $\mathbb{Z}[\zeta_5] = \{a\ + b\,\zeta_5 + c\,\zeta_5^2 + d\,\zeta_5^3 \mid a, b, c, d \in \mathbb{Z}\}$ is the ring of algebraic integers of $\mathbb{Q}(\zeta_5)$. $\mathbb{Z}[\zeta_5]$ is considered as the 4-dimensional lattice $L = [1, \zeta_5, \zeta_5^2, \zeta_5^3]$. The fundamental parallelogram is 4-dimensional: it has 16 points, 32 edges, 24 faces and 8 cubes (see Fig. 2). $\mathbb{Z}[\zeta_8] = \{a\ + b\,\zeta_8 + c\,\zeta_8^2 + d\,\zeta_8^3 \mid a, b, c, d \in \mathbb{Z}\}$ is the ring of algebraic integers of $\mathbb{Q}(\zeta_8)$. $\mathbb{Z}[\zeta_8]$ is the 4-dimensional lattice $L = [1, \zeta_8, \zeta_8^2, \zeta_8^3]$. In Fig. 3 the fundamental parallelogram for the lattice is shown.

In [5], the norms of vectors in 4-dimensional lattices are defined. In $\mathbb{Z}[\zeta_5]$, for $z = a\ + b\,\zeta_5 + c\,\zeta_5^2 + d\,\zeta_5^3$,

$$\mathcal{N}(z)^2 = 2a^2 + 2b^2 + 2c^2 + 2d^2 - ab - ac - bc - bd - cd - da\ . \tag{4}$$

In $\mathbb{Z}[\zeta_8]$, for $z = a\ + b\,\zeta_8 + c\,\zeta_8^2 + d\,\zeta_8^3$,

$$\mathcal{N}(z)^2 = 2a^2 + 2b^2 + 2c^2 + 2d^2\ . \tag{5}$$

3 New Frobenius Method for Hyperelliptic Curves

3.1 5th Roots of Unity

In this section, we show that when $p \equiv 1 \pmod 5$, the coefficients of a Frobenius expansion can be represented using a 5th root of unity $\rho = \frac{-1+\sqrt{5}}{4} + i\frac{\sqrt{5+\sqrt{5}}}{2\sqrt{2}}$. We begin by proving that for the curve (2), $\varphi \in \mathbb{Z}[\rho]$ is well defined.

Lemma 1. *Let* $p \equiv 1 \pmod 5$. *On* X_1 *in (2), Frobenius map* φ *satisfies* $\varphi \in \mathbb{Z}[\rho]$.

Proof. Let $\mathbb{Q}(\rho) = \{u_0 + u_1\rho + u_2\rho^2 + u_3\rho^3 \mid u_i \in \mathbb{Q}\}$. It is well-known that the set of all algebraic integers in $\mathbb{Q}(\rho)$ is $\mathbb{Z}[\rho] = \{c_0 + c_1\rho + c_2\rho^2 + c_3\rho^3 \mid c_i \in \mathbb{Z}\}$ (see [18]). Since $\mathbb{Q}(\rho) \subset End(X_1) \otimes \mathbb{Q} \cong \mathbb{Q}(\varphi)$ and $[\mathbb{Q}(\rho) : \mathbb{Q}] = [\mathbb{Q}(\varphi) : \mathbb{Q}] = 4$ (see [19]), $\mathbb{Q}(\rho) = \mathbb{Q}(\varphi)$. Thus, $\varphi \in \mathbb{Q}(\rho)$. Since φ satisfies the characteristic polynomial $\varphi^4 + a_1\varphi^3 + a_2\varphi^2 + a_1 p\varphi + p^2$, φ is also an algebraic integer. Therefore, $\varphi \in \mathbb{Z}[\rho]$. $\square$

Lemma 2. *Let* $p \equiv 1 \pmod 5$ *and* $s \in \mathbb{Z}[\rho]$. *There exist* $r,\ t \in \mathbb{Z}[\rho]$ *such that* $s = t\varphi + r$ *and* $\mathcal{N}(r) \leq \sqrt{\frac{5p}{2}}$.

Proof. By Lemma 1, φ can be written as $a + b\rho + c\rho^2 + d\rho^3$ for $a,\ b,\ c,\ d \in \mathbb{Z}$. Note that $\mathcal{N}(\varphi) = \sqrt{2p}$. Let $s = s_0 + s_1\rho + s_2\rho^2 + s_3\rho^3$ for $s_i \in \mathbb{Z}$. Then, there exists a quotient $x = x_0 + x_1\rho + x_2\rho^2 + x_3\rho^3$ $(x_i \in \mathbb{Q})$ such that $s = \varphi \cdot x$. If we represent s as $(s_0\ s_1\ s_2\ s_3)$, we get

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = A \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = A^{-1} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} ,$$

where

$$A = \begin{pmatrix} a & -d & -c+d & -b+c \\ b & a-d & -c & -b+d \\ c & b-d & a-c & -b \\ d & c-d & b-c & a-b \end{pmatrix} .$$

To find a quotient in $\mathbb{Z}[\rho]$, set

$$t = (\lfloor x_0 \rceil\ \lfloor x_1 \rceil\ \lfloor x_2 \rceil\ \lfloor x_3 \rceil\ ) ,$$

where $\lfloor z \rceil$ means the nearest integer to z. Then, put

$$r = s - t\varphi = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} - A \begin{pmatrix} \lfloor x_0 \rceil \\ \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \\ \lfloor x_3 \rceil \end{pmatrix} .$$

The largest norm between points in the fundamental parallelogram in $\mathbb{Z}[\rho]$ is $\sqrt{10}$. Thus, the largest norm between points in the fundamental parallelogram in $\varphi\mathbb{Z}[\rho]$ is less than or equal to $\sqrt{10p}$ since $\mathcal{N}(\varphi \cdot x) = \sqrt{p}\mathcal{N}(x)$ [5]. Thus, any lattice point of $\mathbb{Z}[\rho]$ has its nearest point of $\varphi\mathbb{Z}[\rho]$ with the distance less than or equal to $\frac{\sqrt{10p}}{2} = \sqrt{\frac{5p}{2}}$. □

Theorem 1. *Let $p \equiv 1 \pmod 5$ and $s \in \mathbb{Z}[\rho]$. Then we write*

$$s = \sum_{i=0}^{k} r_i \varphi^i , \tag{6}$$

where $r_i \in \mathbb{Z}[\rho]$, $\mathcal{N}(r_i) \leq \sqrt{\frac{5p}{2}}$ and $k \leq \lceil 2 \log_p \mathcal{N}(s) \rceil$. (The expansion of s is finite, not periodic.)

Proof. Let $s_0 = s$. By Lemma 2, $s_0 = s_1\varphi + r_0$. Recursively, $s_j = s_{j+1}\varphi + r_j$. Then,

$$\begin{aligned} s &= s_0 \\ &= s_1\varphi + r_0 \\ &= (s_2\varphi + r_1)\varphi + r_0 = s_2\varphi^2 + r_1\varphi + r_0 \\ &= \left(\sum_{i=0}^{j} r_i\varphi^i\right) + s_{j+1}\varphi^{j+1} , \end{aligned} \tag{7}$$

with $\mathcal{N}(r_i) \le \sqrt{\frac{5p}{2}}$ for $0 \le i \le j$. Using the triangular inequality, we get

$$\begin{aligned}
\mathcal{N}(s_{j+1}) &\le \frac{\mathcal{N}(s_j) + \mathcal{N}(r_j)}{\sqrt{p}} \\
&\le \frac{\mathcal{N}(s_j) + \sqrt{\frac{5p}{2}}}{\sqrt{p}} = \frac{\mathcal{N}(s_j)}{\sqrt{p}} + \sqrt{\frac{5}{2}} \\
&\le \frac{\mathcal{N}(s_{j-1})}{\sqrt{p}^2} + \sqrt{\frac{5}{2}}\left(1 + \frac{1}{\sqrt{p}}\right) \\
&\le \frac{\mathcal{N}(s_0)}{\sqrt{p}^{j+1}} + \sqrt{\frac{5}{2}}\sum_{i=0}^{j}\left(\frac{1}{\sqrt{p}}\right)^i \\
&\le \frac{\mathcal{N}(s_0)}{\sqrt{p}^{j+1}} + \sqrt{\frac{5}{2}} \cdot \frac{\sqrt{p}}{\sqrt{p}-1} \ . \qquad (8)
\end{aligned}$$

Now if $j \ge \lceil 2\log_p \mathcal{N}(s_0)\rceil - 1$, then

$$\frac{\mathcal{N}(s_0)}{\sqrt{p}^{j+1}} \le 1 \ . \qquad (9)$$

We see

$$1 + \sqrt{\frac{5}{2}} \cdot \frac{\sqrt{p}}{\sqrt{p}-1} < \sqrt{\frac{5p}{2}} \ , \qquad (10)$$

since $p \equiv 1 \pmod 5$ is prime, i.e., $p \ge 11$. By (8), (9) and (10), we get

$$\mathcal{N}(s_{j+1}) < \sqrt{\frac{5p}{2}} \ .$$

Setting $s_{j+1} = r_{j+1}$ in (7), we get the expansion (1) with k at most $\lceil 2\log_p \mathcal{N}(s)\rceil$. □

For example, consider $p = 11$ and the curve $X_1 : y^2 = x^5 + 1$. Its Frobenius endomorphism can be written as $\varphi = -1 - 2\rho - 2\rho^2 - 4\rho^3$. The number of such remainders r in Lemma 2 is $11^2 = 121$. We can represent $r = a + b\rho^2$, where $a, b \in \{1,\ \rho,\ 2,\ 2\rho,\ 1 \pm \rho,\ 1 \pm 2\rho,\ 2 \pm \rho,\ 2 + 2\rho,\ 1 + 3\rho,\ 3 + \rho,\ 3 + 2\rho,\ 3 + 3\rho\}$. We can expand 37 as follows:

$$37 = (1 - \rho - \rho^2)\varphi^3 + (\rho + 3\rho^2 + \rho^3)\varphi^2 + (2 + \rho + \rho^2 + \rho^3)\varphi - 2 - \rho + \rho^2 \ .$$

3.2 8th Roots of Unity

In this section, we show that when $p \equiv 1 \pmod 8$, the coefficients of a Frobenius expansion can be represented using an 8th root of unity $\gamma = \frac{1+i}{\sqrt{2}}$. The proof of the following lemma is similar to that of Lemma 1.

Lemma 3. *Let $p \equiv 1 \pmod 8$. On X_2 in (3), the Frobenius map φ satisfies $\varphi \in \mathbb{Z}[\gamma]$.*

Lemma 4. *Let $p \equiv 1 \pmod 8$ and $s \in \mathbb{Z}[\gamma]$. There exist $r,\ t \in \mathbb{Z}[\gamma]$ such that $s = t\varphi + r$ and $\mathcal{N}(r) \le \sqrt{2p}$.*

Proof. By Lemma 3, φ can be written as $a + b\,\gamma + c\,\gamma^2 + d\,\gamma^3$ for $a,\ b,\ c,\ d \in \mathbb{Z}$. Let $s = s_0 + s_1\gamma + s_2\gamma^2 + s_3\gamma^3$ for $s_i \in \mathbb{Z}$. Then there exists a quotient $x = x_0 + x_1\gamma + x_2\gamma^2 + x_3\gamma^3$ $(x_i \in \mathbb{Q})$ such that $s = \varphi \cdot x$. If we represent s as $(s_0\ s_1\ s_2\ s_3)$, we get

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = B \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = B^{-1} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} ,$$

where

$$B = \begin{pmatrix} a & -d & -c & -b \\ b & a & -d & -c \\ c & b & a & -d \\ d & c & b & a \end{pmatrix} .$$

To find a quotient in $\mathbb{Z}[\gamma]$, set

$$t = (\lfloor x_0 \rceil\ \lfloor x_1 \rceil\ \lfloor x_2 \rceil\ \lfloor x_3 \rceil\ ) ,$$

where $\lfloor z \rceil$ means the nearest integer to z. Then, put

$$r = s - t\varphi = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} - B \begin{pmatrix} \lfloor x_0 \rceil \\ \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \\ \lfloor x_3 \rceil \end{pmatrix} .$$

The proof of $\mathcal{N}(r) \le \sqrt{2p}$ is similar to that of Lemma 2. □

Theorem 2. *Let $p \equiv 1 \pmod 8$ and $s \in \mathbb{Z}[\gamma]$. Then we write*

$$s = \sum_{i=0}^{k} r_i \varphi^i , \tag{11}$$

where $r_i \in \mathbb{Z}[\gamma]$, $\mathcal{N}(r_i) \le \sqrt{2p}$ and $k \le \lceil 2 \log_p \mathcal{N}(s) \rceil$. (The expansion of s is finite, not periodic.)

Proof. Omitted. □

For example, consider $p = 17$ and the curve $X_2 : y^2 = x^5 + 2x$. Its Frobenius endomorphism can be written as $\varphi = -2\gamma - 3\gamma^2 + 2\gamma^3$. The number of such remainders r in Lemma 4 is $17^2 = 289$. We can expand 37 as follows:

$$37 = (2 + 2\gamma)\varphi^2 + (1 + 2\gamma + 2\gamma^3)\varphi + 2 + 2\gamma .$$

4 Algorithms

In this section, we present practical algorithms that perform scalar multiplication in hyperelliptic curves with genus 2 using our new expansion method. First, we explain a well-known algorithm that uses the Frobenius map over $\mathbb{F}_{p^n}$, i.e., the hyperelliptic curve version [5, 6] of the Kobayashi-Morita-Kobayashi-Hoshino algorithm [20, 21], which we call hereafter Algorithm KMKH. Then we show how these algorithms can be adapted to use our new expansion method.

The following algorithm is the hyperelliptic curve version of Algorithm KMKH, and it consists of three steps. The first step is the Frobenius expansion step of m, which uses Lange's expansion algorithm [5]. In the second step, the length of the expansion is reduced to n using $\varphi^n(D) = D$.[2] From now on, subscripts are used to denote array indices, and superscripts with parentheses are used to denote bit positions, where the least significant bit is regarded as the 0-th bit.

```
**Algorithm 1**
**Input**: integer $m$, divisor $D$
**Output**: divisor $Q = mD$
**Step 1**: Frobenius expansion of $m$ [5]
    $i \leftarrow 0$, $c_0 \leftarrow m$, $c_1 \leftarrow 0$, $c_2 \leftarrow 0$, $c_3 \leftarrow 0$.
    while ($c_0 \neq 0$ or $c_1 \neq 0$ or $c_2 \neq 0$ or $c_3 \neq 0$) do
        $d \leftarrow \lfloor c_0/p^2 \rceil$, $u_i \leftarrow c_0 - dp^2$, where $\lfloor z \rceil$ means the nearest integer to $z$.
        $c_0 \leftarrow c_1 - a_1pd$, $c_1 \leftarrow c_2 - a_2d$, $c_2 \leftarrow c_3 - a_1d$, $c_3 \leftarrow -d$,
            where $a_1, a_2$ are from the characteristic polynomial $\varphi^4 + a_1\varphi^3 + a_2\varphi^2 +$
$pa_1\varphi + p^2 = 0$.
        $i \leftarrow i + 1$.
    od.
**Step 2**: optimization of the Frobenius expansion using $\varphi^n(D) = D$ [20, 21]
    $r_i \leftarrow u_i + u_{i+n} + u_{i+2n} + u_{i+3n} + u_{i+4n}$ for $0 \leq i < n$.[3]
**Step 3**: scalar multiplication
    $D_i \leftarrow \varphi^i(D)$ for $0 \leq i < n$.
    $Q \leftarrow \mathcal{O}$.
    for $j \leftarrow \max_{i=0}^{n-1} \lceil \log_2 |r_i| \rceil - 1$ to 0 do
        $Q \leftarrow 2Q$.
        for $i = 0$ to $n - 1$ do
            if ($r_i > 0$ and $r_i^{(j)} = 1$) then $Q \leftarrow Q + D_i$.
            else if ($r_i < 0$ and $(-r_i)^{(j)} = 1$) then $Q \leftarrow Q - D_i$.
        od.
    od.
```

[2] Note that it is possible to first reduce m modulo $(\varphi^n - 1)/(\varphi - 1)$, and then to apply the first step, which produces an expansion with smaller coefficients [22, 23]. In [5], the same approach is taken. However, we don't use this approach, since it does not seem to bring a significant speed-up to the overall performance, i.e., it reduces the number of bits in each coefficient at most by two.

[3] According to Lemma 8.2 in [5], the expansion length can be slightly greater than $4n$.

The above algorithm can be modified to use the endomorphism ρ as well as the Frobenius map as follows.

```
Algorithm 2
Input: integer $m$, divisor $D$
Output: divisor $Q = mD$
Step 1: expansion of $m$ using the new expansion method
  $i \leftarrow 0,\ s_0 \leftarrow m,\ s_1 \leftarrow 0,\ s_2 \leftarrow 0,\ s_3 \leftarrow 0$.
  while ($s_0 \neq 0$ or $s_1 \neq 0$ or $s_2 \neq 0$ or $s_3 \neq 0$) do
    $\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \leftarrow A^{-1} \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}, \begin{pmatrix} u_{i,0} \\ u_{i,1} \\ u_{i,2} \\ u_{i,3} \end{pmatrix} \leftarrow \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} - A \begin{pmatrix} \lfloor x_0 \rceil \\ \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \\ \lfloor x_3 \rceil \end{pmatrix}, \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} \leftarrow \begin{pmatrix} \lfloor x_1 \rceil \\ \lfloor x_2 \rceil \\ \lfloor x_2 \rceil \\ \lfloor x_3 \rceil \end{pmatrix}$
    $i \leftarrow i + 1$.
  od.
Step 2: optimization of the expansion using $\varphi^n(D) = D$ [20, 21]
  $r_{ij} \leftarrow u_{i,j} + u_{i+n,j} + u_{i+2n,j} + u_{i+3n,j} + u_{i+4n,j}$ for $0 \leq i < n, 0 \leq j \leq 3$.[4]
Step 3: scalar multiplication
  $D_i \leftarrow \varphi^i(D)$ for $0 \leq i < n$.
  $Q \leftarrow \mathcal{O}$.
  for $k \leftarrow \max_{i,j} \lceil \log_2 |r_{ij}| \rceil - 1$ to 0 do
    $Q \leftarrow 2Q$.
    for $i = 0$ to $n - 1$ do
      for $j = 0$ to 3 do
        if ($r_{ij} > 0$ and $r_{ij}^{(k)} = 1$) then $Q \leftarrow Q + \rho^j(D_i)$.
        else if ($r_{ij} < 0$ and $(-r_{ij})^{(k)} = 1$) then $Q \leftarrow Q - \rho^j(D_i)$.
      od.
    od.
  od.
```

Note that Algorithm 1 also can be modified to use the endomorphism γ. We omit the description of this modification, since it can be constructed similarly to Algorithm 2.

Now we compare the number of divisor operations in Algorithm 2 with that of Algorithm 1. See Table 1. Note that in Algorithm 2, the number of coefficients is quadrupled, but the size of each coefficient is reduced by a fourth root order. Hence, the number of divisor additions is approximately the same. But, the number of divisor doublings is reduced to a quarter, which is the main improvement of our algorithm. Although Algorithm 2 needs $3n$ computations of ρ or γ maps, the required time for these operations is negligible. Finally, we remark that the required memory to store the expansion coefficients (r_i or $r_{i,j}$) and divisors D_i is approximately the same for the two algorithms.

[4] According to Theorem 1, the expansion length can be slightly greater than $4n$.

Table 1. Comparison of the number of divisor operations

	Algorithm 1	Algorithm 2
expansion length (after optimization)	n	n
number of coefficients	n	$4n$
number of bits in each coefficient	$\max_i \lceil \log_2 \lvert r_i \rvert \rceil$ $\approx 2 \log_2 p$	$\max_{i,j} \lceil \log_2 \lvert r_{ij} \rvert \rceil$ $\approx (\log_2 p)/2$
average number of divisor additions[a]	$\approx n \log_2 p$	$\approx n \log_2 p$
number of divisor doublings	$\approx 2 \log_2 p$	$\approx (\log_2 p)/2$
number of Frobenius maps	$n-1$	$n-1$
number of ρ or γ maps	0	$3n$

[a] the total number of bits / 2

Table 2. Implemented fields and curves

curve	p	n	irreducible binomial	curve equation	order (bits)	endo-morphism
curve 1	1021	17	$f(x) = x^{17} - 2$	$y^2 = x^5 + 2$	267	ρ
curve 2	8191	13	$f(x) = x^{13} - 2$	$y^2 = x^5 + 1$	268	ρ
curve 3	8161	17	$f(x) = x^{17} - 2$	$y^2 = x^5 + 1$	416	ρ
curve 4	457	19	$f(x) = x^{19} - 2$	$y^2 = x^5 + 5x$	318	γ
curve 5	761	19	$f(x) = x^{19} - 2$	$y^2 = x^5 + 2x$	336	γ

5 Performance Analysis

In this section, we compare the performance of scalar multiplication algorithms described in the previous section. For the underlying fields, we consider only finite fields $\mathbb{F}_{p^n}$ that have irreducible binomials $f(x) = x^n - w$ as their field polynomials. This is a generalization of optimal extension fields [24, 25], and enables us to implement underlying field operations efficiently.

The fields and curves that we have implemented are shown in Table 2. We can calculate the orders of some Jacobian groups and the characteristic polynomials of the Frobenius maps φ with the help of the program made by Lange [26] that uses MAGMA [27].

Table 3 shows the timings for scalar multiplications on a 2.66GHz Pentium 4 CPU with 512MB RAM using Visual C++ 6.0 compiler. For reference, we have also shown the results for the NAF scalar multiplication algorithm. As shown in Table 3, our method improves the throughput by 15.6–28.3%. According to our experiments, the time required for an expansion is equivalent to only a few divisor additions.

We remark that our comparison could be done on more optimized versions of Algorithms 1 and 2, i.e., we could use NAF's for each coefficient r_i or $r_{i,j}$, a Joint Sparse Form [28], and an on-line precomputation method such as Lim and Hwang's algorithm [29]. Note that in these cases the gains are expected to be greater than those of Table 3, since these optimizations reduce only the number

Table 3. Timings for scalar multiplications (msec)

curve	NAF	Algorithm 1	Algorithm 2	gain[a]
curve 1	382.81	127.19	109.06	16.6%
curve 2	250.00	86.40	67.35	28.3%
curve 3	722.34	194.38	168.12	15.6%
curve 4	543.91	149.37	120.63	23.8%
curve 5	575.79	157.96	130.79	20.8%

[a] throughput increase of Algorithm 2 over Algorithm 1

of divisor additions while preserving the number of doublings unchanged (i.e., the portions of doublings in the overall computations become greater). However, our method does not seem to give much improvement in the divisor-known-in-advance case, since one can reduce the required number of on-line doublings by precomputing some of the doublings in the off-line precomputation stage.

References

[1] Park, Y., Jeong, S., Lim, J.: Speeding up point multiplication on hyperelliptic curves with efficient-computable endomorphisms. In: Advances in Cryptology-EUROCRYPT 02. Volume 2332 of LNCS., Springer-Verlag (2002) 197–208 152, 153, 154, 155

[2] Koblitz, N.: Hyperelliptic cryptosystems. Journal of Cryptology **1** (1989) 139–150 152, 153, 154

[3] Koblitz, N.: CM-curves with good cryptographic properties. In: Advances in Cryptology-CRYPTO 91. Volume 576 of LNCS., Springer-Verlag (1991) 279–287 152

[4] Günther, C., Lange, T., Stein, A.: Speeding up the arithmetic on Koblitz curves of genus two. In: Selected areas in cryptography SAC 2001. Volume 2012 of LNCS., Springer-Verlag (2001) 106–117 152

[5] Lange, T.: Efficient arithmetic on hyperelliptic Koblitz curves. Ph.D. thesis, University of Essen (2001) 152, 153, 154, 157, 158, 161

[6] Choie, Y., Lee, J.: Speeding up the scalar multiplication in the Jacobians of hyperelliptic curves using Frobenius map. In: Progress in Cryptology - INDOCRYPT 2002. Volume 2551 of LNCS., Springer-Verlag (2001) 285–295 153, 161

[7] Gallant, R., Lambert, R., Vanstone, S.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Advances in Cryptology-CRYPTO 2001. Volume 2139 of LNCS., Springer-Verlag (2001) 190–200 153

[8] Park, T., Lee, M., Park, K.: New Frobenius expansions for elliptic curves with efficient endomorphisms. In: Information Security and Cryptology - ICISC 2002. Volume 2587 of LNCS., Springer-Verlag (2002) 264–282 153

[9] Gaudry, P., Hess, F., Smart, N.: Constructive and destructive facets of weil descent on elliptic curves. Journal of Cryptology **15** (2002) 19–46 153

[10] Menezes, A. J., Wu, Y. H., Zuccherato, R. J.: An elementary introduction to hyperelliptic curves. Technical Report CORR 96-19, University of Wateroo (1996) 153

[11] Hartshorne, R.: Algebraic Geometry. Springer-Verlag (1977) 154

[12] Mumford, D.: Tata Lectures on Theta I. Birkhäuser (1983) 154

[13] Cantor, D.: Computing in the Jacobian of a hyperelliptic curve. Mathematics of Computation **48** (1987) 95–101 154

[14] Stein, A.: Sharp upper bounds for arithmetics in hyperelliptic function fields. Technical Report CORR 99-23, University of Wateroo (1999) 154

[15] Buhler, J., Koblitz, N.: Lattice basis reduction, Jacobi sums and hyperelliptic cryptosystems. Bull. Austral. Math. Soc. **58** (1998) 147–154 154

[16] Duursma, I., Gaudry, P., Morain, F.: Speeding up the discrete log computation on curves with automorphisms. In: Advances in Cryptology -ASIACRYPT 99. Volume 1716 of LNCS., Springer-Verlag (1999) 103–121 154

[17] Tate, J.: Endomorphisms of abelian varieties over finite fields. Invent. Math. **2** (1966) 134–144 155

[18] Hardy, G., Wright, E.: An Introduction to the Theory of Numbers. 3rd edn. Oxford University Press (1954) 157

[19] Shimura, G.: Abelian Varieties with Complex Multiplication and Modular Functions. Princeton university press (1998) 157

[20] Kobayashi, T., Morita, H., Kobayashi, K., Hoshino, F.: Fast elliptic curve algorithm combining Frobenius map and table reference to adapt to higher characteristic. In: Advances in Cryptology-EUROCRYPT 99. Volume 1592 of LNCS., Springer-Verlag (1999) 176–189 161, 162

[21] Kobayashi, T.: Base-ϕ method for elliptic curves over OEF. IEICE Trans. Fundamentals **E83-A** (2000) 679–686 161, 162

[22] Solinas, J.: An improved algorithm for arithmetic on a family of elliptic curves. In: Advances in Cryptology-CRYPTO 97. Volume 1294 of LNCS., Springer-Verlag (1997) 357–371 161

[23] Solinas, J.: Efficient arithmetic on Koblitz curves. Designs, Codes and Cryptography **19** (2000) 195–249 161

[24] Bailey, D., Paar, C.: Optimal extension fields for fast arithmetic in public key algorithms. In: Advances in Cryptology-CRYPTO 98. Volume 1462 of LNCS., Springer-Verlag (1998) 472–485 163

[25] Bailey, D., Paar, C.: Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. Journal of Cryptology **14** (2001) 153–176 163

[26] Lange, T.: (FrobSelf) Available at http://www.itsc.ruhr-uni-bochum.de/tanja/KoblitzC.html#progs. 163

[27] MAGMA Group: (MAGMA V2.10 –The Magma computational algebra system) http://magma.maths.usyd.edu.au/magma/. 163

[28] Solinas, J.: Low-weight binary representations for pairs of integers. Technical Report CORR 2001-41, University of Watoroo (2001) 163

[29] Lim, C., Hwang, H.: Speeding up elliptic scalar multiplication with precomputation. In: Information Security and Cryptology-ICISC 99. Volume 1787 of LNCS., Springer-Verlag (1999) 102–119 163

Adaptive Protocol for Entity Authentication and Key Agreement in Mobile Networks

Muxiang Zhang

Verizon Communications Inc.
40 Sylvan Road, Waltham, MA 02451, USA
muxiang.zhang@verizon.com

Abstract. This paper investigates authentication and key agreement protocols running in the dynamic environment in mobile networks. Following the multi-party simulatability approach, we present a formal security model for symmetric-key based authentication and key agreement protocols in the mobile setting. Within this model, we unveil the vulnerability of the authentication and key agreement protocol adopted by Universal Mobile Telecommunication System (UMTS), an emerging standard for third generation (3G) mobile communications. The vulnerability allows an adversary to re-direct user traffic to an unintended network. It also allows an adversary to use authentication data obtained from a corrupted network to impersonate all other networks. In this paper, we present an authentication and key agreement protocol which addresses both security and operational issues involved with UMTS authentication and key agreement. The protocol, called *AP-AKA*, retains the framework of UMTS authentication and key agreement but eliminates synchronization between a mobile station and its home network. Within our formal model, we prove the security of AP-AKA under the assumption of secure message authentication code and pseudorandom function family. For AP-AKA, we also show that the corruption of one operator's network will not jeopardize other operators' networks.

1 Introduction

Authentication and key agreement protocols are basic building blocks for secure communications. To date, such protocols are mostly studied in two basic settings, i.e., the two-party setting and the three-party setting. In both settings, the security association between communicating entities is fixed; they either share a secret data or possess some data generated by a party which is trusted by all entities. This, however, may not be the case in mobile communications. In a mobile environment, the communication service is typically supplied by multiple regional networks, each called a serving network (SN) and operated under a different administration. A user, represented by a mobile station (MS), is usually associated with one serving network, called the home network of the user. At times, however, the user may roam into a serving network (called a foreign network) outside of his home network. This leads to dynamic association between communicating entities. Due to the dynamic association, protocol execution may vary with user movement.

To deal with the dynamic association problem, it is natural to consider the use of public-key cryptographic techniques. This approach, however, may require the construction of large (or even, global) public key infrastructure in order to support global roaming. Furthermore, authentication and key agreement protocols are usually running at low layers, e.g., link layer. In such low layers, user traffic is not routable and will not be passed to higher layers prior to successful authentication. This makes it difficult for the user to verify public key certificates during protocol execution. There are also concerns on the processing power of mobile stations and the bandwidth consumption in exchanging public keys of large size.

In this paper, we restrict our attention to symmetric-key based authentication and key agreement protocols. Over the last decade, dozens of protocols specifically designed for mobile networks have been proposed, e.g., [4, 9, 10, 11, 29, 30, 32, 34]. Most of the proposed protocols were designed based on ad-hoc approaches (i.e., breaking and fixing) and have been found containing various flaws [15, 19, 20, 27]. This phenomenon is well exemplified by the authentication and key agreement protocol adopted by the Global System for Mobile (GSM) communications [21]. The GSM authentication and key agreement protocol (hereafter called GSM AKA) is based a challenge-response mechanism, that is, the user proves his identity by providing a response to a time-variant challenge raised by the network. During roaming, the foreign network requests a set of authentication data (called triplets) from the home network so that the home network is not on-line involved in every authentication process. GSM AKA is simple and has several nice features. Nevertheless, a number of flaws have been uncovered over time [12, 24, 25, 26, 28].

Although limitations of ad-hoc approaches have been widely recognized and protocols achieving provable security have been proposed in other settings, e.g., [6, 7, 23], there are rarely known protocols which possess the simple framework of GSM AKA and are also provably secure. Given this background, there is no surprise that even the Universal Mobile Telecommunication System (UMTS), an emerging standard for third generation (3G) mobile communications, has adopted an authentication and key agreement protocol (hereafter called UMTS AKA) based on ad-hoc approaches. Using a formal method known as BAN logic [18], the designers have claimed [2] that "the goals of the protocol as they are stated in ... are met by the protocol". As has been pointed out by many authors, e.g., [16, 22, 31], this type of analysis is likely to find out bugs in the protocol, but unlikely to provide meaningful security guarantees. This motivates us to investigate protocols such that they can be implemented efficiently in mobile networks and their security can be analyzed within the framework of modern cryptography.

In this paper, we present a formal model of security for symmetric-key based authentication and key agreement protocols in the mobile setting. Our model is a combination of Shoup's [33] formal model of security for authenticated key exchange in the two-party setting and in the three-party setting. We also formulates the definition of entity authentication. Within our model, We analyze the (in)security of UMTS AKA. The flaw of UMTS AKA allows an adversary to re-direct user traffic to an unintended network. The re-direction attack represents a real threat since the security levels provided by different networks are not always the same. In addition, the re-direction attack could cause billing problem as service rates offered by different networks are not always the same, either. Furthermore, the flaw of UMTS AKA also allows an adversary to use authentication data obtained from a corrupted network to impersonate all other networks.

To fix the aforementioned flaw and provide further enhancement on UMTS AKA, we develop an authentication and key agreement protocol which addresses both security and operational issues involved with UMTS AKA. The protocol, called *AP-AKA*, specifies a sequence of six flows. Each flow defines a message type and format sent or received by an entity. How the flows are actually carried out and under what conditions entities accept or reject are dependent on the execution environment. In certain scenarios, only two or three flows are carried out in a protocol execution, while in certain other scenario, all the six flows are carried out in the protocol execution. Dependent on the execution environment, entities have the flexibility of adaptively selecting flows for execution. It is in this sense that we call AP-AKA an adaptive protocol. This is in difference with a conventional two-party or three-party authentication and key agreement protocol in which entities usually execute all the flows specified by the protocol. It is shown that the adaptability helps to optimize the efficiency of AP-AKA both in the home network and in foreign networks.

Within our formal model, we analyze the security of AP-AKA against a powerful adversary having full control of the communication channels between a user and a network. We first consider the case that the adversary cannot corrupt honest networks to obtain authentication data of users. In this case, we prove that AP-AKA is a secure authentication and key agreement protocol under the assumption of secure message authentication code and pseudorandom function family. Then, we consider the case that the adversary can adaptively corrupt networks to obtain authentication data of users. In this case, we show that the corruption of a network could only affect those users who either subscribe to or roam into the corrupted network. The adversary could not impersonate an uncorrupted network to establish communication session with a user whose home network is not corrupted. Thus, the impact of network corruption is drastically lowered on AP-AKA in comparison with UMTS AKA.

Related work on security models. Bellare and Rogaway [6] proposed the first formal model of security for symmetric-key based authentication and key distribution protocols in the two-party setting. In their subsequent work [7] Bellare and Rogaway extended their model to analyze key distribution protocols in the three-party setting. The Bellare-Rogaway model was also adapted to treat public-key based protocols by Blake-Wilson et al. [16, 13]. In [8] Bellare, Canetti, and Krawczyk presented a different approach to formal model of security for authentication and key exchange protocols. This approach stems from the multi-party simulatability tradition [5]. Shoup [33] extended this approach and proposed a formal model of security for key exchange protocols in the three-party setting. Based on the simulatability approach, Boyko, et al. [17] recently proposed a formal model of security for password-authenticated key exchange protocols.

2 Preliminaries

Let $\{0,1\}^n$ denote the set of binary strings of length n and $\{0,1\}^{\leq n}$ denote the set of binary strings of length at most n. For two binary strings s_1 and s_2, the concatenation of s_1 and s_2 is denoted by $s_1 \| s_2$. A real-valued function $\epsilon(k)$ of non-negative integers is called *negligible* (in k) if for every $c > 0$, there exists $k_0 > 0$ such that $\epsilon(k) \leq 1/k^c$ for all $k > k_0$.

Let $X = \{X_k\}_{k \geq 0}$ and $Y = \{Y_k\}_{k \geq 0}$ be sequences of random variables, where X_k and Y_k take values in a finite set S_k. For a probabilistic polynomial-time algorithm D that outputs 0 or 1, we define the *distinguishing advantage* of D as the function

$$\mathbf{Adv}^{dist}_{X_k,Y_k}(D) = |Pr(D(X_k) = 1) - Pr(D(Y_k) = 1)|.$$

If for every probabilistic polynomial-time algorithm, the distinguishing advantage is negligible in k, we say that X and Y are *computationally indistinguishable.*

Let $G : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^s$ denote a family of functions and let $\mathcal{U}(d,s)$ denote the family of all functions from $\{0,1\}^d$ to $\{0,1\}^s$. For a probabilistic polynomial-time oracle machine A, the *prf-advantage* of A is defined as

$$\mathbf{Adv}^{prf}_G(A) = |Pr(g \overset{R}{\leftarrow} G : A^g = 1) - Pr(g \overset{R}{\leftarrow} \mathcal{U}(d,s) : A^g = 1)|,$$

where $g \overset{R}{\leftarrow} G$ denotes the operation of randomly selecting a function g from the family G. We associate to G an insecurity function:

$$\mathbf{Adv}^{prf}_G(t,q) = \max_{A \in \mathcal{A}(t,q)} \mathbf{Adv}^{prf}_G(A),$$

where $\mathcal{A}(t,q)$ denotes the set of adversaries that make at most q oracle queries and have running time at most t. Assume that d and s are polynomials in k. If for every probabilistic polynomial-time oracle machine A, $\mathbf{Adv}^{prf}_G(A)$ is negligible in k, then we say that G is a *pseudorandom* function family.

A *Message Authentication Code* is a family of functions F of $\{0,1\}^k \times Dom(F)$ to $\{0,1\}^l$, where $Dom(F)$ denotes the domain of F. In this paper, $Dom(F) = \{0,1\}^{\leq L}$. For $K \in \{0,1\}^k$ and $M \in \{0,1\}^{\leq L}$, let $\sigma = F(K,M)$. We refer to σ as the tag or MAC of M. For the security of F, we will use the notion of security against chosen message attacks, as defined in [23]. An adversary, called a forger in this context, is a probabilistic polynomial-time algorithm which has access to an oracle that computes MACs under a randomly chosen key K. We define the *mac-advantage* of an adversary A, denoted by $\mathbf{Adv}_F^{mac}(A)$, as the probability that $A^{F(K,)}$ outputs a pair (σ, M) such that $\sigma = F(K, M)$, and M was not a query of A to its oracle. We associate to F an insecurity function,

$$\mathbf{Adv}_F^{mac}(t, q) = \max_{A \in \mathcal{A}(t,q)} \mathbf{Adv}_F^{mac}(A),$$

where $\mathcal{A}(t, q)$ denotes the set of adversaries that make at most q oracle queries and have running time at most t. If for every polynomially bounded adversary A, $\mathbf{Adv}_F^{mac}(A)$ is negligible in k, we say that F is a *secure* message authentication code.

3 Security Model

Our security model consists of two systems, an ideal system and a real system. Security is based on simulatability of adversaries in the two systems. The ideal system follows Shoup's [33] formal model of security for authenticated key exchange in the two-party setting. The real system is adapted from Shoup's [33] formal model of security for authenticated key exchange in the three-party setting. In this section, we assume that the communication channels within and between networks are secured. In Section 6, we consider the case in which adversaries can adaptively corrupt such communication channels.

3.1 The Ideal System

In the ideal system, we do not distinguish between foreign networks and home networks; they are all called serving networks, or simply, networks. Each user or network is called an entity. Communications are between user entities and network entities. In the ideal system, there is an *adversary*, who plays a game by issuing a sequence of operations to a *ring master*. All the adversary can do is create and connect entity instances according to some rules, whereby entity instances obtain random session keys from the ring master. User instances that are connected share a common session key. The adversary learns no more information about the session key other than that is leaked through its use. Basically, the ideal system describes the service that an authentication and key agreement protocol is supposed to provide. For further motivations on the ideal system, we refer the readers to [33].

Let E_i, indexed $i = 1, 2, \ldots$, denote all the entities. Each entity E_i may have several *instances* I_{ij}, $j = 1, 2, \ldots$. In the ideal system, the adversary can execute six types of operations:

(**initialize entity**, $i, role_i, ID_i$): This operation specifies an identity ID_i, along with a value $role_i \in \{0, 1\}$. When $role_i = 0$, the entity is initialized as a user, denoted by U_i; otherwise, the entity is initialized as a network, denoted by N_i. The identity ID_i may be an arbitrary bit string that has not been assigned to another entity. The execution of this operation produces a record of the form (`initialize entity`, $i, role_i, ID_i$).

(**initialize entity instance**, i, j, PID_{ij}): This operation specifies an entity instance I_{ij}, along with a partner identity PID_{ij}. The entity U_i or N_i must have been previously initialized, but I_{ij} should not haven been previously initialized. After the execution of this operation, the instance I_{ij} is *active* and remains active until the execution of either an *abort session* or *start session* operation on I_{ij}. This operation produces a record of the form (`initialize entity instance`, i, j, PID_{ij}).

(**abort session**, i, j): This operation specifies an initialized entity instance I_{ij}. The record for this operation is (`abort session`, i, j).

(**start session**, $i, j, connection\_assignment[, key]$): This operation specifies an active entity instance I_{ij}, along with a connection assignment that specifies how the session key K_{ij} is generated for I_{ij}. This operation produces a record (`start session`, i, j). There are three possible connection assignments: (`create`, i', j'), (`connect`, i', j'), and `compromise`. Each connection assignment is selected under certain rules. The optional *key* field is present only if the *connection_assignment* is `compromise`. The session key K_{ij} is determined according to *connection_assignment* as follows:

– (`create`, i', j'): the ring master generates a random bit string for K_{ij}. This assignment is legal if: (i) $I_{i'j'}$ is a previously initialized instance *compatible* with I_{ij}, that is, $PID_{ij} = ID_{i'}, PID_{i'j'} = ID_i$ and $role_i \neq role_{i'}$; and (ii) the connection assignment (`create`, i', j') was not made before, either on I_{ij} or on other instances. When the *start session* operation completes, we say that I_{ij} is isolated for $I_{i'j'}$.
– (`connect`, i', j'): the ring master sets K_{ij} equal to $K_{i'j'}$. This assignment is legal if $I_{i'j'}$ has been isolated for I_{ij}. When the *start session* operation completes, $I_{i'j'}$ is no longer isolated.
– `compromise`: the ring master sets K_{ij} equal to *key*.

(**application**, f): This operation specifies a function f which is applied to the session keys $\{K_{ij}\}$ and a random input R. After the execution of this operation, the ring master gives the adversary $f(R, \{K_{ij}\})$. This operation produces a record of the form (`application`, $f, f(R, \{K_{ij}\})$).

(**implementation**, *comment*): This operation allows the adversary to add a *comment* to the transcript. It is recorded as (`implementation`,*comment*).

As the adversary executes operations in the ideal system, a transcript logging his activities is constructed. For an adversary $\mathcal{A}^*$, we use $\mathcal{T}_{\mathcal{A}^*}$ to denote the transcript.

3.2 The Real System

We now describe the real system which models the operations executed by a powerful real-world adversary who has full control of the communication channels between a user and a network. The real-world adversary can deliver messages out of order and to unintended recipients, concoct messages of her own choosing, and start up arbitrary number of sessions between a user and a network. Moreover, it is the adversary that drives everything forward; users and networks only follow the instructions of the real-world adversary. The objective of the real-world adversary is to defeat the goal set for an authentication and key agreement protocol. The adversary is not only interested in recovering session keys, but also interested in establishing communication sessions between unintended entities.

As in the ideal system, the real-world adversary initializes an entity by executing the operation (**initialize entity**, $i, role_i, ID_i$). This operation is logged in the transcript as (`initialize entity`, $i, role_i, ID_i$). Likewise, the real-world adversary initializes an entity instance via the operation (**initialize entity instance**, i, j, PID_{ij}). The record for this operation is (`initialize entity instance`, i, j, PID_{ij}). Unlike in the ideal system, however, entities and entity instances in the real system are not just placeholders; they have internal states and can generate messages of their choosing.

When an entity is initialized as a network N_i with identity ID_i, a protocol-specific *initialization routine* is started to initialize the internal state of N_i and to establish service agreement with all previously initialized networks. To do so, the ring master in the real system provides to the initialization routine a list of previously initialized networks, the routine selects a network, say $N_{i'}$, from the list and sends the message (ID_i, *service agreement*) to $N_{i'}$. Upon receipt of the message, $N_{i'}$ creates two mailboxes, $InMail_i$ and $OutMail_i$, for N_i. Correspondingly, N_i also creates two mailboxes, $InMail_{i'}$ and $OutMail_{i'}$ for $N_{i'}$. Both N_i and its instances can write messages into $OutMail_{i'}$ and read messages from $InMail_{i'}$; the adversary is not allowed to access the mailboxes. Both *read* and *write* are atomic operations. When a message is written into $OutMail_{i'}$ by N_i or its instances, the ring master takes out the message and delivers it to the mailbox $InMail_i$ of $N_{i'}$. Through the ring master and the mailboxes, we model a secure channel between N_i and $N_{i'}$. All the instances of N_i and $N_{i'}$ share this secure channel.

In the real system, each user is associated with a network, i.e., the home network, which has been previously initialized. To specify the association between a user and his home network, we assume that the identity of the user is prefixed by identity of the home network. When an entity is initialized as a user U_i with identity ID_i, a protocol-specific *registration routine* is started to initialize the internal state of U_i and to register U_i's identity with its home network $N_{i'}$. During the registration, the ring master generates a random bit string, denoted by K_i, and delivers K_i to both U_i and $N_{i'}$.

An entity instance I_{ij} is a state machine. After starting in some initial state, the state of I_{ij} may be updated by a message delivered by the adversary. The message-delivery operation takes the form (**deliver message**, $i, j, type, InMsg$), where a *type* is assigned to the message $InMsg$. In response to the message delivery, I_{ij} updates its state, generates a response message $OutMsg$, and reports its $status$. The response message and status information are given to the adversary. There are three possible statuses:

continue: I_{ij} is prepared to receive another message.

accept: I_{ij} is finished and has generated a session key SK_{ij}.

reject: I_{ij} is finished but refuses to generate a session key.

If the *status* is not `continue`, I_{ij} is no longer *active*. Dependent on the status, this operation generates one or two records. The first record is (`implementation, deliver message`, i, j, *type*, $InMsg$, $OutMsg$, *status*). The second record is (`start session`, i, j) if *status* =`accept`, or (`abort session`, i, j) if else.

In the real system, the adversary may also execute the application operation (**application**, f). This is the same operation as described in the ideal system, except that the actual session keys $\{SK_{ij}\}$ are used. In addition, the random input R is independent of any bits used by entities or entity instances during initialization and during the execution of the protocol. The application operation produces the record (`application`, $f, f(R, \{SK_{ij}\})$).

3.3 Definition of Security

Security is based on simulatability of adversaries in the two different systems. There are two requirements:

Completion. For every efficient real-world adversary that faithfully delivers messages between two compatible entity instances, both entity instances accept and share the same session key.

Simulatability. For every efficient real-world adversary $\mathcal{A}$, there exists an ideal-world adversary $\mathcal{A}^*$ such that their transcripts $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{T}^*_{\mathcal{A}}$ are computationally indistinguishable.

The above requirements are essentially the same as those given by Shoup [33]. In Shoup's security model, the connection assignment `create` is always legal. So there may be multiple instances which are isolated for an initialized instance. In our model, we add restrictions to the connection assignment `create` to ensure that at most one instance is isolated for an initialized instance.

4 Analysis of UMTS AKA

In UMTS AKA, each user U and his home network, denoted by HN, share a secret key K and certain cryptographic algorithms. In addition, the home network HN maintains a counter SQN_{HN} for each individual subscriber, and the mobile station maintains a counter SQN_U for the user U. The initial values for SQN_{HN} and SQN_U are zero. The cryptographic algorithms shared between HN and U include two message authentication codes $f1, f2$ and three key generation functions $f3, f4, f5$. For generic requirements as well as an example algorithm set for these cryptographic algorithms, refer to [3].

UMTS AKA consists of two phases. The first phase specifies the distribution of authentication data, called authentication vectors, from the home network to the serving network. The second phase specifies the authentication and key agreement procedure between the user and the serving network. When the protocol is executed in the home network or when the serving network has unused authentication vectors for the user, the first phase is not executed. For a concise description of the two phases, let's assume that a user U roams into a foreign network SN which does not have unused authentication vectors for the user. In this scenario, the protocol can be described as follows:

1. $SN \rightarrow HN$: *authentication data request*, ID_U
2. $HN \rightarrow SN$: *auth. data response*, $\{(RAND, XRES, CK, IK, AUTH), \ldots\}$
   where $XRES = f2_K(RAND)$, $CK = f3_K(RAND)$,
   $IK = f4_K(RAND)$, $AUTN = SQN \oplus AK \| AMF \| MAC$,
   $AK = f5_K(RAND)$, $MAC = f1_K(SQN || RAND || AMF)$.
3. $SN \rightarrow U$: *user authentication request*, $RAND, AUTH$
4. $U \rightarrow SN$: *user authentication response*, $RES = f2_K(RAND)$, or
   user authentication reject, $CAUSE$

The serving network SN starts the protocol by sending *authentication data request* to the home network HN, where *authentication data request* is the *type* of the message. Upon receipt of the message, HN sends back a batch of authentication vectors. For each authentication vector, HN generates a random number, $RAND$, and a sequence number, SQN, from the counter SQN_{HN} maintained for U. Then HN computes an expected response, RES, a cipher key, CK, an integrity key, IK, an authentication token, $AUTH$, and increases the counter SQN_{HN} by 1.

After receiving the authentication vectors from HN, SN selects one of them and stores the rest in its database. Then SN sends to the user $RAND$ and $AUTN$ from the selected authentication vector. The user computes AK and retrieves SQN from the received $AUTH$. Next, the user verifies the correctness of the MAC included in $AUTH$. If the verification fails, the user rejects. Otherwise, the user further verifies if the sequence number SQN is in the correct range, i.e., $SQN > SQN_U$. If not, the user sends a *synchronization failure* message to SN and rejects. Otherwise, the user sets SQN_U to SQN, sends a response back to SN, and accepts.

Note that the identity of the serving network is not included in the cryptographic functions in UMTS AKA, although it is broadcasted in a public channel. UMTS network supports two types of authentication mechanisms: "an authentication mechanism using an authentication vector delivered by the user's home network to the serving network, and a local authentication mechanism using the integrity key established between the user and the serving network during previous execution of the authentication and key establishment procedure" (see [1]). It is clear that the UMTS AKA as described above achieves mutual authentication between the user and the home network, not mutual authentication between the user and the serving network, since an authentication vector could be ordered by any serving network. The local authentication mechanism, on the other hand, intends to achieve mutual authentication between the user and the serving network. According to [1], one of the objectives for UMTS AKA is to defeat the so-called false base station attack [28]. In such an attack, an adversary intends to impersonate a serving network by exploiting a device having the functionality of a base station. In the following, we show that UMTS AKA is vulnerable to a variant of the false base station attack.

Assume that a user U is in the territory of his home network HN and intends to establishes a communication session with the home network. Also assume that an adversary is operating a false base station which broadcasts the identity of HN. Once the adversary intercepts the connection attempt from the user, the adversary entices the user to camp on the radio channels of the false base station. Then the adversary sends a connection request to a foreign network, say SN, of her choosing. Next, the adversary faithfully delivers message between he user and the foreign network. Authentication will be successful both in the user side and in the foreign network and communication will be protected via established keys. In this way, the adversary can re-direct user traffic to an unintended network. It is worth pointing out that the re-direction attack has practical implications. In UMTS network, data encryption is not mandatory in every serving network, while data integrity is mandatory in all networks. To intercept user traffic, the adversary may re-direct user traffic to a network in which data encryption is either not provided or provided but very weak. The threat of this attack is particularly evident as the user roams into the border of two different networks. It might be argued that the risk could be mitigated if all the networks "trust" each other and use a commonly agreed strong encryption algorithm. Nevertheless, the re-direction attack could cause billing problem; the user is in the territory of his home network but gets charged by a foreign network based on a rate higher than that offered by the home network.

Based on the re-direction attack as described above, we can prove that the protocol UMTS AKA does not meet the requirements of Definition 3.3. The proof of the following theorem can be found in [37].

Theorem 1. *For UMTS AKA, there exists a real-world adversary $\mathcal{A}$ such that for every ideal-world adversary $\mathcal{A}^*$, there is a probabilistic polynomial-time algorithm D,*

$$\mathbf{Adv}^{dist}_{\mathcal{T}_{\mathcal{A}},\mathcal{T}_{\mathcal{A}^*}}(D) = |Pr(D(\mathcal{T}_{\mathcal{A}}) = 1) - Pr(D(\mathcal{T}_{\mathcal{A}^*}) = 1)| \geq 1 - 2^{-\ell},$$

where ℓ is the length of the session key.

Except the re-direction attack, the protocol UMTS AKA may also be attacked in another way. Suppose that a network, say N_i, is corrupted. The adversary can eavesdrop any message sent or received by N_i. In addition, the adversary can concoct an *authentication data request* and sends it to any other network through N_i. In this way, the adversary can obtain authentication vectors for any user, independent of the actual location of the user. The compromised authentication vectors allow the adversary to impersonate all serving networks. So, the corruption of one operator's network jeopardize the entire system. Furthermore, UMTS AKA uses sequence numbers to ensure that authentication vectors are not re-used by a user or by a network. Sequence numbers are generated based on a counter SQN_{HN} maintained in the home network and verified based on another counter SQN_U maintained in the mobile station. When a mismatch happens between the two counters, i.e., $SQN_{HN} < SQN_U$, which may be caused by a failure in the home network, authentication vectors generated by the home network will be rejected by the user. As shown in [36], the need of synchronization between the counter in the mobile station and the counter in the home network incurs considerable difficulty for the normal operation of UMTS AKA.

5 Adaptive Authentication and Key Agreement Protocol

In this section, we present an authentication and key agreement protocol which defeats re-direction attack and drastically lowers the impact of network corruption. The protocol, called AP-AKA, retains the framework of UMTS AKA but eliminates synchronization between the mobile station and its home network. In AP-AKA, the home network does not maintain dynamic states for each individual subscriber. The user can verify whether authentication vectors were indeed ordered by a serving network and were not used before by the serving network. In this section, we specify the operation of AP-AKA in the dynamic execution environment in mobile networks. In the next section, we analyze AP-AKA within our formal model of security.

Assume that each user and his home network share a secret key K of length k and three cryptographic algorithms F, G and H, where F and H are message authentication codes, G is a pseudorandom function family indexed by K. An overview of AP-AKA is described as follows:

1. $SN \rightarrow U$: *user data request*, $FRESH$
2. $U \rightarrow SN$: *user data response*, $RN, U_{MAC} = F_K(FRESH||RN||ID_{SN})$
3. $SN \rightarrow HN$: *authentication data request*, $ID_U, FRESH, RN, U_{MAC}$
4. $HN \rightarrow SN$: *auth. data response*, $\{(RAND, XRES, SK, AUTH), \ldots\}$
   where $XRES = F_K(RAND), SK = G_K(RAND)$,
   $AUTH = idx \| RN_{idx} \| MAC$, $1 \leq idx \leq m$,
   $RN_{idx} = H_K(idx||RN), MAC = F_K(RAND||idx||RN_{idx})$.
5. $SN \rightarrow U$: *user authentication request*, $RAND, AUTH$
6. $U \rightarrow SN$: *user authentication response*, $RES = F_K(RAND)$

The protocol AP-AKA specifies a sequence of six flows. Each flow describes a message *type* and *format* sent or received by an entity. How the flows are carried out and under what conditions entities accept or reject are dependent on the execution environment. In the following, we specify the execution of AP-AKA in various scenarios.

Protocol execution in SN. When a user U roams into a foreign network SN, the protocol execution may be carried out in two different ways.

- If SN has unused authentication vectors for the user, it starts the protocol by executing the fifth flow, i.e., sending *user authentication request* to the user, including $RAND$ and $AUTH$ from the selected authentication vector. After receiving user authentication response, SN compares if $RES = XRES$. If not, SN rejects. Otherwise, SN accepts and the agreed session key is SK from the selected authentication vector.

- If SN does not have unused authentication vectors for the user, then all the six flows will be carried out. SN starts by sending a random number, $FRESH$, to the user. After receiving *user data response*, SN requests authentication data from the home network. The home network verifies the correctness of the received U_{MAC}. If the verification fails, HN sends back a *reject notice* including $ID_U, FRESH$, and RN. Upon receiving the notice, SN rejects. If the verification succeeds, HN sends back a batch of authentication vectors to SN. For each authentication vector, HN generates a random number, $RAND$, and then computes an expected response, $XRES$, a session key, SK, and an authentication token $AUTH$, where SK may be a concatenation of a cipher key and an integrity key. Each authentication vector is indexed by an integer $idx, 1 \leq idx \leq m$, where m is the number of authentication vectors in the batch. After receiving authentication vectors from HN, SN proceeds as in the previous case.

Protocol execution in *HN*. If there are unused authentication vectors for the user, the protocol execution is performed in the same way as in the foreign network. Otherwise, HN executes a three-flow protocol as described below:

1. $HN \rightarrow U$: *user data request*, $FRESH$
2. $U \rightarrow HN$: *user data response*, $RN, U_{MAC} = F_K(FRESH||RN||ID_{HN})$
3. $HN \rightarrow U$: *user authentication request*, $RAND, AUTH$
   $AUTH = 0||RN_0||F_K(RAND||0||RN_0)$, $RN_0 = H_K(0||RN)$.

In the three-flow protocol, HN verifies the correctness of U_{MAC}. If the verification fails, HN rejects. Otherwise, HN computes the session key $SK = G_K(RAND)$ and accepts. In the case of acceptance, HN may also generate a batch of authentication vectors for future use.

User actions. The user U acts as a *responder* in the protocol. When the user receives a *user data request*, it generates a random number, RN, and then replies back with RN and U_{MAC}. When the user receives a *user authentication request*, it retrieves $RAND$ and $AUTN$ from the request and verifies the correctness of the MAC included in $AUTN$. If the verification fails, the user rejects. Otherwise, the user further verifies if the RN_{idx} included in $AUTH$ is acceptable, that is, $RN_{idx} = H_k(idx||RN)$ and RN_{idx} was not used before by the serving network (either a foreign network or a home network). If RN_{idx} is not acceptable, the user rejects. Otherwise the user computes the session key $SK = G_K(RAND)$ and then proceeds as follows:

- If $idx > 0$, the user updates its internal state, replies back with RES, and accepts.
- If $idx = 0$, the user updates its internal state and accepts. In this case, the user does not send back a response.

To facilitate fast verification of RN_{idx}, the user maintains usage information on RN_{idx}. The usage information is used to verify that the authentication vector containing RN_{idx} was indeed ordered by the serving network and was not used before by the serving network. This helps to defeat re-direction attack. It also helps to prevent replay attack which exploits previously used authentication vectors. In Appendix A, we provide a fast verification scheme for RN_{idx}.

In AP-AKA, we assume that the user has knowledge of the identity of the serving network. Otherwise, an adversary can broadcast a fraudulent network identity and re-directs the user's connection to an unintended serving network. In [38], we describe practical methods for the user to detect fraudulent network identities. Assume that the home network generates a batch of m authentication vectors in response to each *authentication data request* and all the authentication vectors will be used eventually. Then the average number of flows carried out in a foreign network is $\eta_{SN} = (6 + 2(m - 1))/m = 2 + 4/m$. E.g., $\eta_{SN} = 2.8$ when $m = 5$. The average number of flows carried out in HN is $\eta_{HN} = 3$ if HN does not generate authentication vectors; otherwise $\eta_{HN} = (3 + 2m)/(m + 1)$. For UMTS AKA, $\eta_{SN} = (4 + 2(m - 1))/m = 2 + 2/m$ and $\eta_{HN} = 2$. So AP-AKA is slightly costly in traffic than UMTS AKA.

6 Analysis of AP-AKA

In AP-AKA, the user authenticates the network through the MAC included in *user authentication request*. If the verification succeeds, the user in ensured that the network is either the home network or a foreign network authorized by the home network to provide him service. In addition, by verifying the RN_{idx} included in $AUTH$, the user is assured that the authentication vector was ordered by the serving network and was not used before by the serving network. Thus, an adversary can not entice the user to establish communication session with unintended networks. In the following, we prove the security of AP-AKA under our formal definition of security. Due to space limitation, we omit the proofs of all the theorems given in this section. For details of the proofs, we refer the readers to [37].

6.1 Provable Security without Network Corruption

Assume that the communication channels within and between serving networks (both foreign and home networks) are secured. The adversary has full control of the communication channels between a user and a network, but can not access the mailboxes maintained by a network.

Definition 1. *Let I_{ij} be an entity instance in the real system. A stimulus on I_{ij} is a message such that the status of I_{ij} changes from continue to accept after receiving the message.*

Definition 2. *Let $\mathcal{A}$ be a real world adversary and let $\mathcal{T}_{\mathcal{A}}$ be the transcript of $\mathcal{A}$. In the game of $\mathcal{A}$, if the random numbers generated by an entity and its instances are different, we say that $\mathcal{T}_{\mathcal{A}}$ is a collision-free transcript. For every accepted instance I_{ij}, if the stimulus on I_{ij} was output by a compatible instance, we say that $\mathcal{T}_{\mathcal{A}}$ is an authentic transcript.*

Definition 3. *Let $\mathcal{T}_{\mathcal{A}}$ be the transcript of a real-world adversary $\mathcal{A}$ and let $\sigma_1, \sigma_2, \ldots, \sigma_n$ denote all the MACs which are computed under F by entities and entity instances. If $\sigma_i \neq \sigma_j$ for any $i \neq j$, we say that F is collision-resistant in $\mathcal{T}_{\mathcal{A}}$. Similarly, if all the MACs computed under H are different, we say that H is collision-resistant in $\mathcal{T}_{\mathcal{A}}$.*

Lemma 1. *Let $\mathcal{A}$ be a real-world adversary and let $\mathcal{T}_{\mathcal{A}}$ be the transcript of $\mathcal{A}$. Assume that $\mathcal{T}_{\mathcal{A}}$ is collision-free. Also assume that F and H are independent function families and are collision-resistant in $\mathcal{T}_{\mathcal{A}}$. Let $M_{\mathcal{A}}$ denote the event that $\mathcal{T}_{\mathcal{A}}$ is authentic. Then*

$$Pr(\overline{M}_{\mathcal{A}}) \leq n_i(\mathbf{Adv}_F^{mac}(t,q) + \mathbf{Adv}_F^{mac}(t,q)\mathbf{Adv}_H^{mac}(t,q')),$$

where $t = O(T), q = O(3n_i)$, $q' = O(n_i)$, T is the running time of $\mathcal{A}$, and n_i is the number of instances initialized by $\mathcal{A}$.

For authentic and collision-free transcripts, we have the following theorem.

Theorem 2. *Let $\mathcal{A}$ be a real-world adversary and let $\mathcal{T}_{\mathcal{A}}$ be the transcript of $\mathcal{A}$. Assume that $\mathcal{T}_{\mathcal{A}}$ is authentic and collision-free. Also assume that G is a pseudorandom function family, independent of F and H, and F, H are collision-resistant in $\mathcal{T}_{\mathcal{A}}$. Then there exists an ideal-world adversary A^* such that for every distinguisher D with running time T,*

$$\mathbf{Adv}_{\mathcal{T}_{\mathcal{A}},\mathcal{T}_{\mathcal{A}^*}}^{dist}(D) \leq n_e \mathbf{Adv}_G^{prf}(t,q),$$

where n_e is the number of user entities initialized by $\mathcal{A}$ and n_i is the number of instances initialized by $\mathcal{A}$, $t = O(T), q = O(n_i)$.

Based on Lemma 1 and Theorem 2, we have the follow Theorem 3.

Theorem 3. *Assume that G is a pseudorandom function family, F and H are secure message authentication codes, and F, G, and H are independent. Then AP-AKA is a secure authentication and key agreement protocol.*

Theorem 3 indicates that a real-world adversary as described in Section 3.2 could not do any more harm to AP-AKA than an ideal-world adversary could. Since the ideal-world adversary is a benign one; she can only connect compatible instances and can not learn more information about a session key other than that is leaked through its use. Hence, AP-AKA is secure against attacks (including re-direction attack) of the real-world adversary who has full control of the communication channels between a user and a network.

6.2 Impact of Network Corruption

For UMTS AKA, we have discussed that the corruption of a network impairs the security of the entire system. We now examine the impact of network corruption on AP-AKA. When a network, say N_i, is corrupted, an adversary can eavesdrop any message sent or received by N_i. She can also deliver messages to other networks through N_i. Certainly, the adversary could impersonate N_i to establish a communication session with a user roamed into N_i. The adversary could also impersonate any subscriber of N_i. We claim that these are all the adversary could do from a corrupted network; the adversary could not impersonate an uncorrupted network to establish a communication session with a user whose home network is not corrupted. To prove our claim, we need to modify both the ideal system and the real system in our formal model of security. The definition of security, defined in terms of completion and simulatability, will remain the same.

A modification to the real system is that the real-world adversary can execute the operation (**corrupt network**, i) to corrupt an initialized network N_i. This operation is logged in the ideal-world transcript as (`corrupt network`, i). After the execution of this operation, the real-world adversary can access the mailboxes of N_i and obtains authentication vectors sent and received by N_i. In addition, the real-world adversary can also write messages to the mailboxes of N_i.

A modification to the ideal system is that the ideal-world adversary can execute the operation (**corrupt network**, i) to corrupt an initialized network N_i. This operation is logged in the transcript as (`corrupt network`, i). In the ideal system, we also need to modify the rules governing the legitimacy of the connection assignments. For a user instance I_{ij}, the connection assignment `compromise` is legal if either PID_{ij} is assigned to a corrupted network or the home network of U_i is corrupted. For a network instance $I_{i'j'}$, the connection assignment `compromise` is legal if either $N_{i'}$ is corrupted or $PID_{i'j'}$ is assigned to a user whose home network is corrupted.

With the above modifications, we have the following theorem:

Theorem 4. *Assume that G is a pseudorandom function family, F and H are secure message authentication codes, and F, G, and H are independent. Then AP-AKA is an authentication and key agreement protocol secure against network corruption.*

In the ideal system, the adversary $\mathcal{A}^*$ could compromise the session keys of those users who either subscribe to or roam into a corrupted network. The adversary $\mathcal{A}^*$ could not impersonate an uncorrupted network to authenticate a user whose home network is not corrupted. By Theorem 4, the real-world adversary $\mathcal{A}$ could not do more harm than the ideal-world adversary $\mathcal{A}^*$. So the real-word adversary could not impersonate an uncorrupted network to establish a communication session with a user if the user's home network is not corrupted. Therefore, the threat of network corruption is greatly reduced for AP-AKA in comparison with UMTS AKA.

7 Conclusion

This paper investigates authentication and key agreement protocols running in the dynamic environment in mobile networks. Following the multi-party simulatability approach, we present a formal model of security for such protocols. In this model, we first analyze the (in)security of the authentication and key agreement protocol adopted by UMTS. We then present the protocol AP-AKA which defeats re-direction attack and drastically lowers the impact of network corruption. The protocol AP-AKA also eliminates synchronization between a mobile station and its home network. Within our formal model, we prove the security of AP-AKA under the assumption of secure message authentication code and pseudorandom function family. For AP-AKA, we also show that the corruption of one operator's network will not jeopardize other operators' networks.

Acknowledgement

The author would like to thank Dr. Moti Yung for his valuable suggestion and encouragement.

References

[1] 3GPP TS 21.102, 3rd Generation Partnership Project (3GPP); Technical Specification Group (TSG) SA; 3G Security;Security Architecture, version 4.2.0, Release 4, 2001. 174

[2] 3GPP TR 33.902, 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Formal Analysis of the 3G Authentication Protocol", version 3.1.0 (Release 1999). 167

[3] 3GPP TS 21.102, 3rd Generation Partnership Project (3GPP); Technical Specification Group (TSG) SA; 3G Security; Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions $f1$, $f1^*$, $f2$, $f3$, $f4$, $f5$ and $f5^*$; Document 1: General, version 4.2.0, Release 4, 2001. 173

[4] A. Aziz and W. Diffie, Privacy and authentication for wireless local area networks, *IEEE Personal Communications*, vol. 1, 1994, pp. 25-31. 167

[5] D. Beaver, Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority, *Journal of Cryptology*, vol. 4, 1991, pp. 75–122. 169

[6] M. Bellare and P. Rogaway, Entity authentication and key distribution, *Advances in Cryptology - Crypto' 93 Proceedings*, Lecture Notes in Computer Science, vol. 773, Springer-Verlag, 1993. pp. 232-249. 167, 168

[7] M. Bellare and P. Rogaway, Provably secure session key distribution–The three party case, *Proc. 27th ACM Symp. on Theory of Computing*, Las Vegas, NV, USA, May 1995, pp. 57-66. 167, 168

[8] M. Bellare, R. Canetti, and H. Krawczyk, A modular approach to the design and analysis of authentication and key exchange protocols, *Proceedings of 30th Annual ACM Symposium on Theory of Computing*, ACM Press, new York, 1998. 168

[9] M. J. Beller, L.-F. Chang, and Y. Yacobi, Privacy and authentication on a portable communication system, *IEEE Journal on Selected Areas in Communications*, Vol. 11, 1993, pp. 82-829. 167

[10] M. Beller and Y. Yacobi, Fully-fledged two-way public key authentication and key agreement for low-cost terminals, *Electronics Letters*, vol. 29, 1993, pp. 999-1001. 167

[11] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung, The Kryptoknight family of light-weight protocols for authentication and key distribution, *IEEE/ACM Trans. on Networking*, vol.3, 1995, pp. 31-41 167

[12] A. Biryukov, A. Shamir, and D. Wagner, Real time cryptanalysis of the alleged A5/1 on a PC, *Proceedings of Fast Software Encryption Workshop*, April, New York, 2000. 167

[13] S. Blake-Wilson, D. Johnson and A. Menezes, Key agreement protocols and their security analysis, *Cryptography and Coding*, Lecture Notes in Computer Science, vol. 1355, 1997, pp. 30-45. 168

[14] S. Blake-Wilson and A. Menezes, Entity authentication and key transport protocols employing asymmetric techniques", Proceedings of Security Protocols Workshop, 1997. 167, 168

[15] C. Boyd and A. Mathuria, Key establishment protocols for secure mobile communications: A selective survey, *Proceedings of ACISP'98* , Lecture Notes in Computer Science, vol. 1438, 1998, pp. 344-355. 167

[16] C. Boyd and W. Mao, On a limitation of BAN logic, *Advances in Cryptology – Eurocrypt '93 proceedings*, LNCS 765, Springer-Verlag, 1993, pp. 240-247. 167, 168

[17] V. Boyko, P. MacKenzie, and S. Patel, Provably secure password-authenticated key exchange using Diffie-Hellman, *Advances in Cryptology - Eurocrypt'2000 Proceedings*, Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000. pp. 156-171. 169

[18] M. Burrows, M. Abadi, and R. Needham, A logic of authentication, *ACM Transactions on Computer Systems*, vol. 8, 1990, pp.18-36. 167

[19] L. Buttyan, C. Gbaguidi, S. Sttmann, and U. Wilhelm, Extensions to an authentication technique proposed for global mobility network, *IEEE Transactions on Communications*, vol. 48, 2000, pp. 373-376. 167

[20] U. Carlsen, Optimal privacy and authentication on a portable communications system, *Operating Systems Review*, vol. 28, 1994, pp.16-23. 167

[21] European Telecommunications Standards Institute (ETSI), GSM 02.09: Security Aspects, June 1993. 167

[22] V. D. Gligor, L. Gong, R. Kailar, and S. Stubblebine, Logics for cryptographic protocols - virtues and limitations, *Proceedings of the Fourth IEEE Computer Security Foundations Workshop*, New Hampshire, 1991, pp. 219 - 226. 167

[23] S. Goldwasser, S. Micali, and R. Rivest, A Digital signature scheme secure against adaptive chosen message attacks, *SIAM J. Computing*, vol. 17, 1988, pp. 281–308. 167, 169

[24] L. Harn and H. Lin, "Modifications to enhance the security of GSM", *Proceedings of 5th National Conference on Information Security*, Taiwan, pp. 74-76, May 1995. 167

[25] C. H. Lee, M. S. Hwang, and W. P. Yang, Enhanced privacy and authentication for the global system for mobile communications, *Wireless Networks*, vol.5, pp. 231-243, 1999. 167

[26] H. Lin and L. Harn, "Authentication protocols for personal communication system", *Proceedings of ACM SIGCOMM'95*, August 1995. 167

[27] K. M. Martin and C. Mitchell, Comments on an optimized protocol for mobile network authentication and security, *Mobile Computing and Communications Review*, vol. 3, 1999, pp. 37. 167

[28] C. Mitchell, The security of the GSM air interface protocol, Technical Report, RHUL-MA-2001-3, Royal Holloway, University of London, 2001. 167, 174

[29] R. Molva, D. Samfat, and G. Tsudik, Authentication of mobile users, *IEEE Network*, 1994, pp. 26-34. 167

[30] Yi Mu, V. Varadharajan, On the design of security protocols for mobile communications, *Proceedings of ACISP'96*, Lecture Notes in Computer Science, vol. 1172, Springer-Verlag, 1996, pp. 134-145. 167

[31] D. Nessett, A critique of the Burrows, Abadi and Needham logic, *ACM SIGOPS Operating Systems Review*, vol. 24, 1990, pp.35-38. 167

[32] C. Park, K. Kurosawa, T. Okamoto and S. Tsujii, On key distribution and authentication in mobile radio networks, *Advances in Cryptology-Eurocrypt'93 Proceedings*, Lecture Notes in Computer Science, vol. 765, 1993, pp. 461-465. 167

[33] V. Shoup, On formal models for secure key exchange, *Proceedings of the Sixth Annual ACM Conference on Computer and Communications security (invited talk)*, 1999. 167, 169, 170, 173

[34] M. Tatebayashi, N. Matsuzaki and D. B. J. Newman, Key distribution protocol for digital mobile communication systems, *Advances in Cryptology-Crypto '89 Proceedings*, Lecture Notes in Computer Science, vol. 435, 1989, pp. 324-334. 167

[35] W. Tzeng and C. Hu, Inter-protocol interleaving attacks on some authentication and key distribution protocols, *Information Processing Letters*, vol. 69, 1999, pp. 297-302.

[36] M. Zhang, A robust authentication and key agreement protocol for third-generation wireless networks, *Proceedings of the IASTED Int. Conf. on Communications and Computer Networks*, November 4-6, 2002, Cambridge, Massachusetts, pp. 1-6. 176

[37] M. Zhang, Provably-secure enhancement on 3GPP authentication and key agreement protocol, *Cryptology ePrint Archive*, Report 2003/092. 175, 178

[38] M. Zhang and Y. Fang, Security analysis and enhancements on 3GPP authentication and key agreement protocol, submitted to *IEEE Trans. Wireless Communications*, 2003. 178

A Verification of RN_{idx}

In AP-AKA, each number RN_{idx} can only be used once. For convenience, we refer to each number RN_{idx} as a nonce. To support fast verification of RN_{idx}, the user maintains a list of unused nonces for every visited network. Each list is pointed by the identity of the network. Assume that the home network generates a batch of m authentication vectors in response to each *authentication data request*. After receiving *user data request* from a network N_i, the user replies back with RN and U_{MAC}. Then the user computes a sequence of nonces $RN_r = H_K(r, RN)$, $r = 0, 1, \ldots, m$, and adds the computed nonces to the list pointed by ID_{N_i}. When the user verifies a nonce RN_{idx} received from the network N_i, the user only needs to check if RN_{idx} is in the list pointed by ID_{N_i}. If not, RN_{idx} is not acceptable; otherwise, it is acceptable. In the case of acceptance, the user removes RN_{idx} from the list.

Extended Role Based Access Control and Procedural Restrictions

Wook Shin[1], Dong-Ik Lee[1], Hyoung-Chun Kim[2], Jung-Min Kang[2], and Jin-Seok Lee[2]

[1] Department of Information and Communications, Kwang-Ju Institute of Science and Technology, Oryong, Buk, Gwang-Ju, 500-712, Korea
{sunihill,dilee}@kjist.ac.kr

[2] National Security Research Institute, P.O.Box 1, Yuseong, Daejeon, 305-600, Korea
{khche,jmkang,jinslee}@etri.re.kr

Abstract. The current scheme of access control judges the legality of each access based on immediate information without considering associate information hidden in a series of accesses. Due to the limitation, access control systems do not efficiently limit attacks consist of allowed operations. For trusted operating system developments, we extended RBAC and added procedural constraints to refuse those attacks. With the procedural constraints, the access control of trusted operating systems can discriminate attack trials from normal behaviors. Also, extended RBAC keeps the principle of least privilege and separation of duty more precisely. This paper shows the specification of the extended concept and model, and presents simple analysis results.

1 Introduction

Trusted Computing Base (TCB) is the totality of a computer system's protection mechanisms [1], [2], [3]. TCB enforces a unified security policy to guarantee confidentiality, integrity, and availability of the system.

The development of a trusted operating system embodies the concept of TCB by placing a security kernel as a central mechanism and providing other security services such as authentication, encryption, audit, and so on. The security kernel implements the reference monitor concept. The reference monitor mediates all accesses under access control policies. The access control mechanism of the security kernel is essential to trusted OS developments.

Traditionally, general UNIX-compatible operating systems have controlled accesses under Discretionary Access Control (DAC) policy. However, DAC does not support a sufficient level of security. Early trusted OS developments adopted a supplementary policy of Mandatory Access Control (MAC) [4], [5], [6]. Yet, MAC is too strict to be applied to commercial applications due to the lack of flexibility in executions. Therefore, recent trusted OS developments [7], [8], [9] introduce Role Based Access Control (RBAC).

Though various access control policies have been applied, current access control has limitations in its function. The limitation originates from the discreteness of access control information.

At each time of the access, a security kernel gathers information from the subject, the object, and the system environment. However, the relation information between accesses is not considered for making an access decision. Even if the execution of an operation set results in an attack, each element can be allowed separately. In other words, if an attack is composed of permitted operations, and each of the operation does not violate the access rules and the policy of the system, then the access control system does not deny the attack.

Those attacks have been out of range of access control. Instead, applications such as Firewall and Intrusion Detection System (IDS) are adopted for current commercial environments to block those attacks. However, those application-level solutions are not fundamental because of the possibility of by-passing [10].

Gollmann [2] described the reasons for putting security mechanisms into the lower layers: Higher assurance of security and lower performance overheads. Hence, the better solution is placing control mechanisms for those attacks at the kernel level of trusted OS. It corresponds to the motivation of trusted OS developments [11].

In this paper, we discuss the extended concept and model of access control. By this extension, access decisions are not only made from instantly gathered information, but also associated information of accesses. The associated information is stored as the form of a procedure. The access control system tests whether an execution of each operation completes the procedure or not, and block executions which are out of permitted sequence. And, for the extension, we use the notion of RBAC. The concept of the abstraction in RBAC is useful to associate operations and give procedural constraints to the sets of operations.

This paper is organized as follows. In Sects. 2 and 3, the extended RBAC concept and its model are described, respectively. The analysis of the proposed access control is discussed in Sect. 4. Section 5 is the conclusion.

2 Extended RBAC Concepts

2.1 Abstractions in RBAC

Recently developed trusted OSs [7], [8], [9] have introduced RBAC as their major security policy. Providing several advantages [12], [13], RBAC compensates for the lack of ability of the traditional policies.

The advantages stem from abstraction mechanisms. RBAC provides the abstraction based on role entities. A role is the association between a set of users and a set of permissions. The role is useful to manage authorities. The security officer can give or deprive user's permission by assigning or withdrawing the role. Also, the defined roles can have some constraints or hierarchical organizations for more convenient authority managements [12].

For the extension in Sect. 2.3, it is necessary to investigate the characteristics of the RBAC abstraction in this Section. The abstractions in RBAC can be

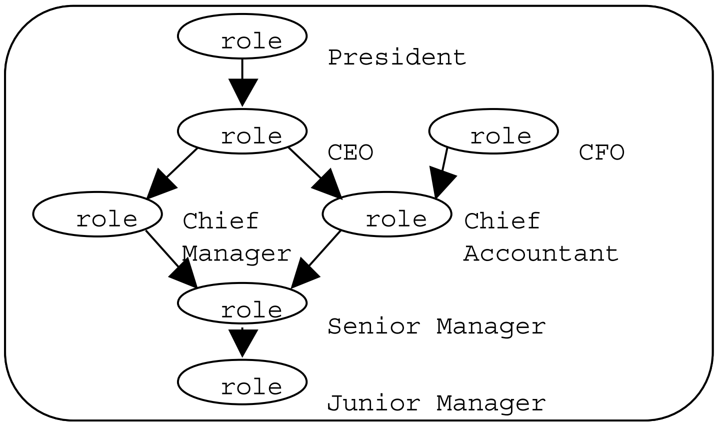


Fig. 1. Role hierarchy which is originated from user organization [17]

classified as subject-abstraction and object-abstraction based on the inherent property of an access; an access is the interaction between a subject and an object.

The subject-abstraction is organizing users as roles based on users' real-world organization. Some roles reflect the real positions and responsibilities of users such as 'Doctor', 'Nurse' or 'Patient', so that the administrator can grasp comparatively clear image on whom to be assigned. Several examples are presented in previous researches [12], [13], [14], [15].

On the other hand, the object-abstraction is specifying system resources as roles. Such roles as 'Accounting', 'Auditing' are the result of the object-abstraction. They consist of primitive operations of a target system, and give ideas of what are the functions of the roles.

The abstractions can constitute a systematic structure, called 'hierarchy'. Semantically or functionally related roles are associated each other with inheritance relationships, and a role can be defined in the hierarchy. The hierarchy improves reusability of role definitions and helps reflecting real-world user organizations. Finally, it reduces administrative costs. Moffett [16] categorized constructing methods in the hierarchy constitutions as aggregation, generalization, and organization. Fig. 1 shows a result of the organization. Such advantages from the abstraction in RBAC are similar to the advantages of the object-oriented approach of software engineering.

Also, roles are useful to constrain subjects and objects. Separation of duty (SOD) relationships can be defined between roles to prohibit two or more roles being assigned to a particular user. More various conditions such as access time, access domain can be applied to limit accesses properly.

Although the concept of roles has not been biased in one direction, the abstractions of RBAC have been studied partially. Most of noted researches on role hierarchy have focused on the subject-abstraction. Context information for building role hierarchies has been mainly extracted from user organizations. The examples in the researches [12], [16], [18] show how the hierarchies are constructed based on user organizations. Fig. 1 presents one of the example.

On the other hand, describing resources as abstracted forms and controlling accesses are important for system implementations such as trusted OS. The extended access control concept which is mentioned in Sect. 1 can be accomplished by constraining and organizing permissions as the form of procedures. Hence, we concentrate upon the object-abstraction of RBAC in this paper.

2.2 Motives of the Extension

Access control information has gained from users and objects at each time of an access. To check permissions and make an access decision, the information is used instantly. However, the instant decision restricts the functions of access control system. Assume that permissions on file accesses are defined as $\{$*fexec*, *fread*, *fwrite*, *flink*, *funlink*$\}$ in a system. When a user tries to remove a link of the file, a traditional access control system gets information from the user and the file, and checks if the acquired roles of the user have *funlink* on the file. At the next moment, if the user tries to make a symbolic link to the file, a similar process is performed again. However, the correlation between two actions, unlink and link, is not a matter of concern. The access control system considers the latter action is totally independent to the former, so that the permissions are tested severally. If the user have both link and unlink permissions to the file, the user can repeat link and unlink to the file. Though the repetition is an important signature of the race condition attacks, there is no control.

However, if we use the well-defined object-abstractions with the additional constraints such as an order, repetition information on a set of permissions, we can limit above kinds of accesses and improve the functions of access control.

Assume that we have a set of permissions $\{A, B, C, D\}$. If they are required for a role's task completion, we have assigned all the permissions to the role traditionally. However, if the task needs only one series of permissions like (A, C, B, D), we do not have to allow all combination of permissions to the role. In other words, in the traditional scheme, we have given more authorities to the role than needs. If we check sequences of accesses, we are able to reject the abuse of permissions in an unexpected order. The ordered control helps to keep the principle of least privilege.

Also, we can describe the SOD relationships more accurately. If the sequence of (A, C, B, D) is in SOD relation with the permission order of (B, A, D, C), we can observe the principle of separation of duty more properly.

Additionally, if we adopt the concept of negative permission [2] into RBAC, we can expect that access control systems to refuse attack trials efficiently. It is possible to define a harmful behavior as a 'negative' set of permissions by appending a tag on the set of permissions.

For example, a race condition attack [19], [20] sends a mail and repeats link/unlink actions in sequence. Therefore, if we limit the repetition of link and unlink after a mail-send operation, we can refuse the attack trials.

2.3 The Extended Properties

For the purpose of controlling accesses with an advanced manner, following properties are added to RBAC.

– procedural information: Order and repetition information describe a permission set procedurally.
– positive/negative information: The information indicates whether a set of permissions are harmful of not.
– other constraints: Various information can be added for a precise control such as access time, limited lapse from execution start, owner of the objects, and so on. However, these constraints are not dealt with in this paper for a simplicity.

Although the additional information are useful for the object-abstraction, but not for the subject-abstraction. For example, roles such as 'Doctor', 'Nurse' do not need to have order or repetition information. When those properties are added to the subject-abstraction, we expect that some kinds of disadvantages as follows.

– implementation overhead: The fields for order or repetition will be overhead for the role like 'Bank manager'. Also, we need to write unessential functions to initialize and manipulate those fields, and system will invoke those functions.
– semantical estrangement: It is not semantically clear that the subject-abstraction and the object-abstraction are co-exist in a same layer of 'Roles'. Moreover, confusions from the coexistence can cause erroneous relationships by introducing cycles in a role hierarchy.

To avoid disadvantages, it is considerable to separate the subject-abstraction and the object-abstraction. Roles for the object-abstration can be separated from roles for the subject-abstration in RBAC. We name newly separated roles for the object-abstraction as 'Behaviors'. As a consequence of the separation, the modified concept consists of four main entities of users, roles, behaviors, permissions.

3 Extended RBAC Model

We name the extended access control as Role-Behavior Based Access Control (RBBAC) for convenience' sake. In this section, we describe the model of RBBAC. The description of the RBBAC model is also extended and modified from the standard RBAC model [18].

The RBBAC model specification starts from the core RBBAC model. Core RBBAC just adds an additional abstraction entity, behavior, to RBAC. After the core RBBAC model is described, we introduce the constrained RBBAC model. Constrained RBBAC adds the concept of Procedural Restrictions (PR) to core RBBAC. PR enables a system to control operations elaborately by restraining behaviors to be executed in order.

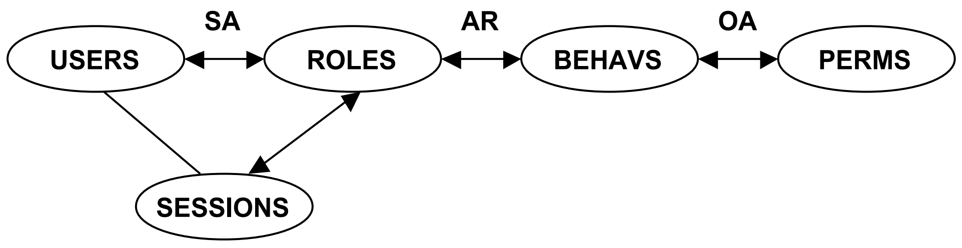


Fig. 2. Extended RBAC entities and relations

3.1 Core RBBAC Model

Fig. 2 shows basic RBBAC elements and relations. The basic elements are users, roles, behaviors, and permissions. Users are subjects of all accesses. Roles are the semantic representatives for users. Permissions are objects of all accesses. Permissions are derived from the combinations of resource objects and access modes. Behaviors are the semantic representatives for permissions as roles do. Sessions are associations between users and roles. Basic relations are subject assignments (SA), access rules (AR), and object assignments (OA). As Fig. 2 shows, SA consists of one-to-one user-session mapping and many-to-many session-role associations. AR is the relation between roles and behaviors. In terms of reference monitor, roles and behaviors are the representatives of the actual access subjects and objects, respectively. Hence, AR is the essential security rules limiting subjects' accesses to objects. OA is the relation between behavior and permissions. Above descriptions are formally described as follows.

Definition 1 (Core RBBAC).

– $USERS, ROLES, BEHAVS, andPERMS$: the set of users, roles, behaviors, and permissions, respectively.
– $SA \subseteq USERS \times ROLES$, the many-to-many user-to-role assignment.
– $assigned\_users : (r : ROLES) \rightarrow 2^{USERS}$, the mapping from a role r onto a set of users,
  • $assigned\_users(r) = \{u \in USERS | (u, r) \in SA\}$
– $OA \subseteq BEHAVS \times PERMS$, the many-to-many privilege-to-behavior assignment.
– $assigned\_permissions : (b : BEHAVS) \rightarrow 2^{PERMS}$, the mapping of a behavior b onto a set of permissions,
  • $assigned\_permissions(b) = \{p \in PERMS | (p, b) \in OA\}$
– SESSIONS: the set of sessions.
– $user\_session(u : USERS) \rightarrow SESSIONS$, the mapping from a user onto a session.
– $session\_role(s : SESSIONS) \rightarrow ROLES$, the mapping from a session s onto a role.
– $AR \subseteq ROLES \times BEHAVS$, the many-to-many role-to-behavior assignment.

- $assigned\_behaviors : (r : ROLES) \rightarrow 2^{BEHAVS}$, the mapping from a role r onto a set of behaviors,
  - $assigned\_behaviors(r) = \{b \in BEHAVS | (r, b) \in AR\}$

On the basis of the above definitions, we define the following function of *avail_session_permissions* to decide whether an access is legal or not. It is the core mechanism of the access decision facilities of an access control system [21].

- $avail\_session\_permissions : (s : SESSIONS) \rightarrow 2^{PERMS}$, the mapping from a session s onto a set of permissions,
  - $avail\_session\_permissions : (s : SESSIONS) \rightarrow 2^{PERMS} =$

$$\bigcup_{b \in assigned\_behaviors(r)} assigned\_permissions(b)$$
$$(, where\ r \in session\_roles(s))$$

At the moment of a user accesses to a permission, access control system confirms the associations between the user's session and the permission, and makes an access decision. In core RBBAC model, the access decision process is consecutive; the system validates roles of the user, behaviors associated to the role, and permissions related with the behaviors. The function, *avail_session_permissions* implicates the consecutive process. The final step of the allowance is that the access control system's confirmation of the targeted permission is in the set of the validated permissions.

3.2 Constrained RBBAC Model

The constrained RBBAC model includes additional constraints of Procedural Restrictions (PR). Basic requirements for giving PR are precise access control and prohibition of dangerous executions. PR provides a solution for the requirements by giving ordering constraints to behaviors and a property for describing whether the ordered behaviors are harmful of not.

Fig. 3 shows the conceptual diagram of constrained RBBAC. We introduce several entities to establish the PR concept as follows.

- Procedural Unit (PU): The extended behavior with an order and an iteration number. If a PU is defined as ('copy', 2, 3), it means that the behavior 'copy' should be thirdly executed in the procedure and executed twice.
- Procedural Constraint (PC): A procedure consists of a set of PUs. And a PC contains an additional property that describes whether the PC is positive or negative. If it is positive, the procedure is executed step by step in a system. If it is negative, the execution of the procedure is blocked before the completion.
- Procedure History (PH): Behavior execution logs. If a permission is accessed, PH is recorded. And the log mainly contains such information as behavior, PU, PC, and sessions.

The following definitions summarize the above descriptions.

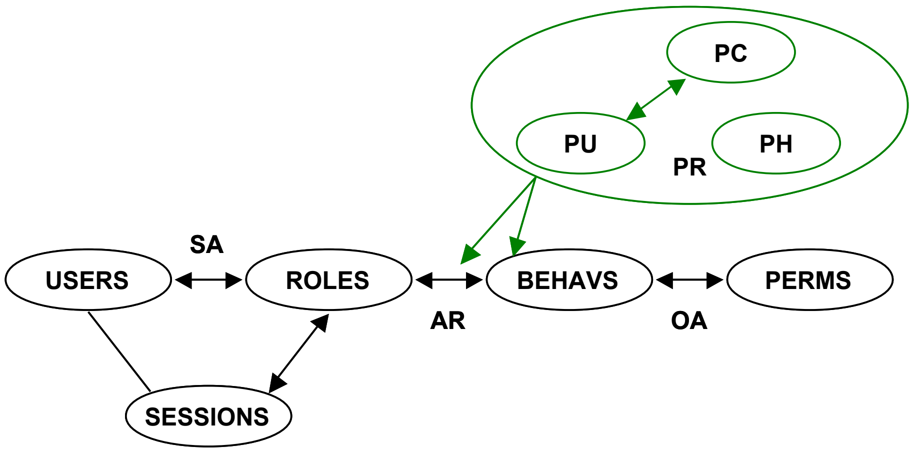


Fig. 3. Extended RBAC entities and relations with procedural restrictions

Definition 2 (Procedural Restrictions).

- $PU \subseteq (BEHAVS \times N \times N)$, the collection of (bs, itnum, onum), where each *bs* is a behavior, and *itnum* and *onum* are natural numbers. *itnum* implies maximum number of repetition for the behavior *bs*. *onum* indicates the order of execution for *bs* in the procedural unit.
- $behavior\_punits : (b : BEHAVS) \rightarrow 2^{PU}$, the mapping from behavior b onto a set of procedural units which include the behavior as a constituent.
- $PC \subseteq (2^{PU} \times prop)$, the collection of pairs (*punits*, *prop*), where each *punits* is a set of procedural units and *prop* is *POSITIVE* or *NEGATIVE*.
- $behavior\_pconstraints : (b : BEHAVS) \rightarrow 2^{PC}$, the mapping from behavior b onto a set of related PC. The returned PC includes PU in which behavior b participates as a constituent.
- $pconstraints\_property(pc) : (pc : PC) \rightarrow \{POSITIVE, NEGATIVE\}$, the mapping of a PC to its property information.
- $order\_of\_behavior(pc, b) : (pc : PC, b : BEHAVS) \rightarrow N$, the mapping from a behavior to a natural number of a PC. The natural number indicates relative execution order of the behavior in the PC.
- $is\_the\_last(pc, b) : (pc : PC, b : BEHAVS) \rightarrow \{TRUE, FALSE\}$, the function returns *TRUE* if the behavior is the last in the PC, otherwise returns *FALSE*.
- $PH \subseteq SESSIONS \times ROLES \times BEHAVS \times N \times PC$, the collection of ($s$, r, b, n, pc), where each s is a session, r is a role, b is a behavior, n is a natural number which indicates order in a procedural constraints, and pc is the procedural constraints.
- $last\_onum\_behavior(s, pc, b) : (s : SESSIONS, pc : PC, b : BEHAVS) \rightarrow N$, returns the number that indicates relative order of the last executed behavior for the session s issued pc.

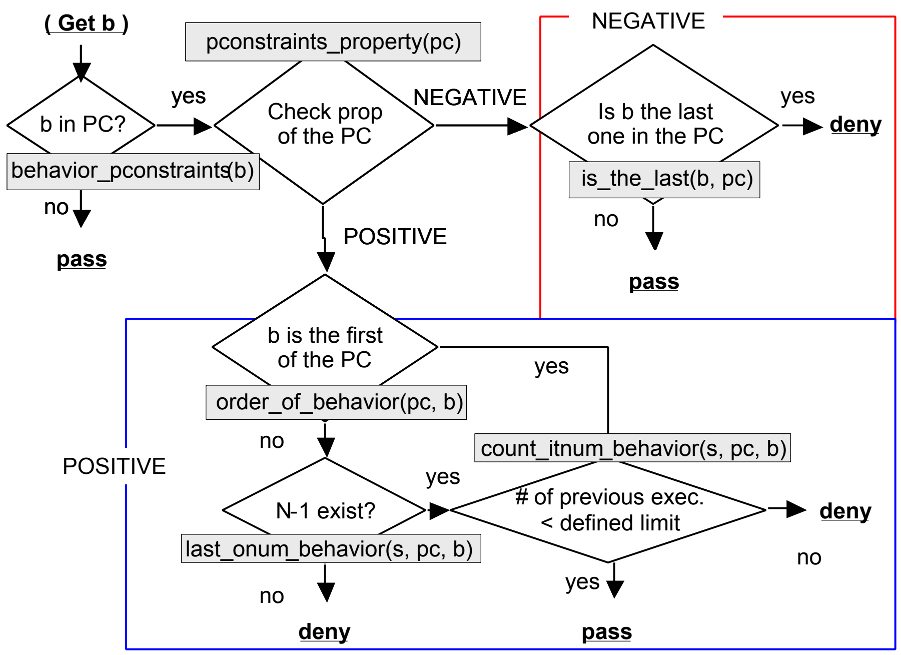


Fig. 4. Testing procedure of an access with procedural restrictions

– $count\_itnum\_behaviors(s, pc, b) : (s : SESSIONS, pc : PC, b : BEHAVS) \rightarrow N$, returns the number of repeated times of the behavior execution for the session s issued pc.

With above definitions for procedural restrictions, an access control system tests each access following the procedure as Fig. 4 shows.

If a user accesses to a permission, access control system gets information of the session and the behavior, and check whether a PC is applied to the behavior or not. If a PC is applied, get the property of the PC. If the property is negative, deny executions of at least one behavior before all the behaviors in the PC are executed. In case of the positive, execute a behavior if all of the preliminary behaviors were executed and if it does not exceed the number of permitted repetitions.

In the decision process, it is possible that a conflict occurs owing to the reusability of behavior definitions. Assume that a behavior participates in two PCs, p1 and p2. If both p1 and p2 are positive PC, it is no problem to permit the executions. If both are negative, and the behavior is the last one of at least one PC, then the execution should be denied. If one PC is positive, and the other is negative, the denial or allowance is decided according to the policy of the system.

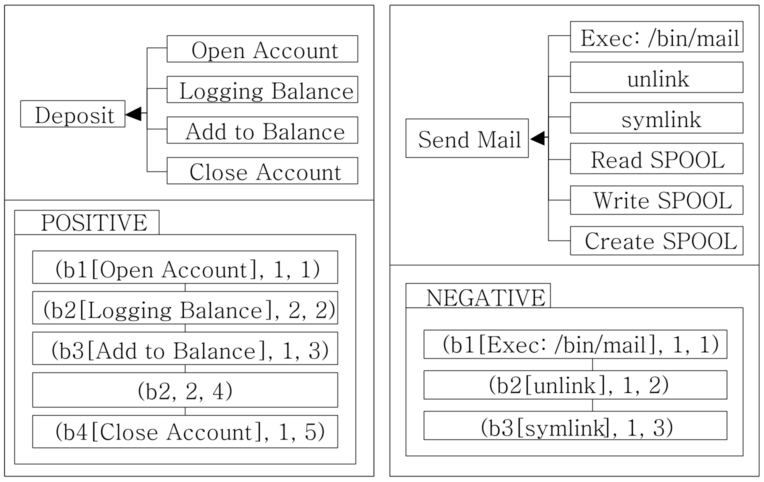


Fig. 5. Example definitions of positive/negative procedure

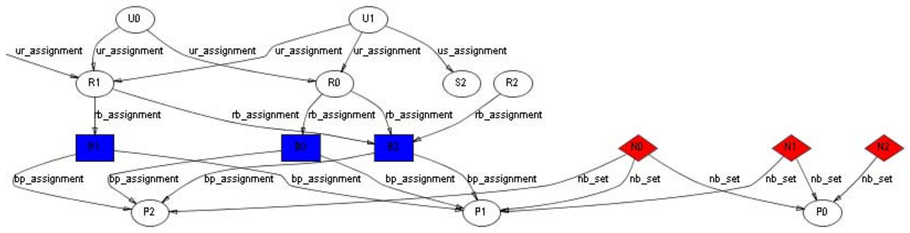


Fig. 6. The diagram of Constrained RBBAC Model

4 Analysis

As mentioned in Sect. 2.2, the more accurate access control is possible with the procedural restrictions. Moreover, the extended model can refuse attacks that consist of legal operations. It has been unable to limit those attack trials in traditional access control of RBAC.

Fig. 5 shows example definitions of behavior with positive and negative PC. In the left figure, 'Deposit' is defined as a positive PC. Without the PC, a malicious user can open an account file and abuse the authority by performing a behavior 'Add to Balance' several times. However, if the PC is applied, the execution is under a restriction: The behavior 'Add to Balance' can be done only once, and the account file is closed.

In the right figure, a set of behavior is defined as a negative PC. Some version of sendmail program has a security flaw leaking administrator's authority. The race condition attack [20] can be used for the intrusion, and the attack tries to do 'unlink' and 'symlink' a temp file to '.rhosts' repeatedly. Therefore, if it is prohibited that the unlink/link repetition by defining negative PC, the system is free from race condition attacks.

Fig. 6 shows a part of the result of modeling constrained RBBAC. For the specification and analysis, we used Alloy [22]. In the figure, the diamonds are negative behaviors and the rectangles are positive behaviors. No positive behaviors are equivalent to negative behaviors.

However, this analysis is not yet complete because the concept of the order is not specified in the diagram. Though we used proof-based approach to specify RBBAC model, we need state-machine based approach to model procedural constraints. The complete specification and analysis of constrained RBBAC will be considered later.

In the extended access control, the main decision process is similar to that of previous RBAC. In RBAC, if a user accesses an object, access control system check whether the permission on the object is associated with the user's acquired role. In the extended method, access control system checks if the behavior have the permission on the object is associated or not with the user's acquired role.

However, the extended access control is different with previous RBAC. In the previous concept, roles are the neutral entities which are placed between users and permissions. It can be interpreted as both, and it is the figuration of the mapping between users and permissions. On the contrary, in the extended concept, roles and behaviors are the abstraction for users and permissions, respectively. They are interpreted as subjects and objects separately. And, in extended RBAC, there are no entities between roles and behaviors, and still associations are exist instead. It is different from the RBAC's visualization of the mapping. In other words, it is possible that the roles can be considered as groups with hierarchy, and behaviors as functions with nested structures.

5 Conclusions

In this paper, we described an extended RBAC concept and model. By the extension, access control not only bases instant access control information but also continuous access information. Thus, access control efficiently limits attack trials which consist of allowed operation. The race condition attack was presented as an example of those attacks trials. The race condition attack is a kind of attack using time-of-check-to-time-of-use (TOCTTOU) [19] vulnerabilities. The attacks exploit inconsistencies of synchronization in multi-user environments and they use ordinary operations. We expect the extended method can deny more attacks based on similar mechanism.

Extended RBAC useful to prevent various known-attacks because the description of attack sequences is supported in procedural restrictions. It means the function of access control is expanded to the coverage of IDSs, though its ability has limitations due to performance overhead. However, for some attacks, access control provides more reliable solutions. Because access control is performed at the kernel level not being by-passed. It is different from IDSs which are executed in the application level.

While we describe the proposed model, several concepts of the standard RBAC model [18] such as inheritance, SSD, and DSD are not included. It is omitted to simplify the extension. We consider it is not difficult to introduce those constraints to the extended models because the extended model was originated from the core RBAC model.

Acknowledgement

We would like to express our gratitude to the late Professor Dong-Ik Lee (1958-2003), from Kwang-Ju Institute of Science and Technology, Korea. He passed away while this paper is being reviewed. We are grateful to him for scientific advice and generous assistance. May he rest in peace.

References

[1] Department of Defense, Department of Defense Trusted Computer System Evaluation Criteria, Department of Defense Standard(DOD 5200.28-STD), Library Number S225, 711, 1985. 184
[2] D. Gollmann, Computer Security, John Wiley & SONS, 1999. 184, 185, 187
[3] E. G. Amoroso, Fundamentals of Computer Security Technology, AT&T Bell Laboratories, Prentice Hall PTR., 1994. 184
[4] Cray Research, UNICOS Multilevel Security (MLS) Feature User's Guide, SG-2111 10.0, Cray Research, Inc., 1990. 184
[5] M. Branstad, H. Tajalli, and F. Mayer, Security issues of the Trusted Mach system, Proc. of 4th Aerospace Computer Security Applications Conference, pp. 362-367, 1998. 184
[6] Flash: `http://www.cs.utah.edu/flux/fluke` 184
[7] P. Loscocco, S. Smalley, Integrating Flexible Support for Security Policies into the Linux Operating System, Proc. of the FREENIX Track: 2001 USENIX Annual Technical Conference (FREENIX '01), 2001. 184, 185
[8] A. Ott, The Rule Set Based Access Control (RSBAC) Linux Kernel Security Extension. 8th Int. Linux Kongress, Enschede, 2001. 184, 185
[9] Trusted Solaris:
`http://wwws.sun.com/ software/solaris/trustedsolaris/index.html` 184, 185
[10] T. Ptacek and T. Newsham, Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection, 1998. 185
[11] D. Baker, Fortresses built upon sand, In Proceedings of the New Security Paradigms Workshop, 1996. 185
[12] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, Role-Based Access Control Models, IEEE Computer, Vol. 29, No. 2, 1996. 185, 186, 187
[13] D. Ferraiolo, J. F. Barkely, and D. R. Kuhn, A Role Based Access Control Model and Reference Implementation within a Corporate Intranet, ACM Transactions on Information Systems Security, Vol.1, No.2, 1999. 185, 186
[14] D. Ferraiolo, J. Cugini, and D. R. Kuhn, Role Based Access Control: Features and Motivations, Proc. of Annual Computer Security Applications Conference, IEEE Computer Society Press, 1995. 186

[15] J. F. Barkley, V. Cincotta, D. F. Ferraiolo, S. Garrvrilla, and D. R. Kuhn, Role Based Access Control for the World Wide Web, NIST 20th National Computer Security Conference, 1997. 186

[16] J. D. Moffett, Control Pinciples and Role Hierarchies, 3rd ACM Workshop on Role Based Access Control (RBAC), pp. 22-23, Oct. 1998. 186, 187

[17] M. Koch, L. V. Mancini, and F. P. Presicce, A Graph-Based Formalism for RBAC, ACM Trancsactions on Information and System Security, Vol. 5, No. 3, pp. 332-365, Aug. 2002. 186

[18] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, Proposed NIST Standard for Role-Based Access Control, ACM Transactions on Information and Systems Security, Vol. 4, No. 3, 2001. 187, 188, 195

[19] M. Bishop and M. Dilger, Checking for Race Conditions in File Access, Computing Systems, Vol. No. 2, 131-152, 1996. 187, 194

[20] [8lgm]-Advisory-20.UNIX.SunOS-sendmailV5.1-Aug-1995.README 187, 193

[21] ITU-T SG/7 & Working Parties: Final text for recommendation X.812 Information Technology-Open Systems interconnection Security framework for open systems: Access control framework, 1995. 190

[22] Alloy: `http://sdg.lcs.mit.edu/alloy/` 194

Layer-Based Access Control Model in the Manufacturing Infrastructure and Design Automation System

Yuan Zhang[1], Moon Jung Chung[1], and Hyun Kim[2]

[1] Computer Science and Engineering Dept.
Michigan State University, USA
[2] ETRI
Taejeon, Korea

Abstract. Design and manufacturing of a complex product requires collaboration among companies. The collaborating companies form a virtual enterprise, which is dynamically organized. In this paper, we propose a Layer-based Access Control (LBAC) system in a collaborative design and manufacturing environment. Our system protects not only data but also processes in a hierarchical distributed workflow environment. The goal is to shield the process from illegal access as well as to protect intellectual properties and trade secrets. Our proposed approach protects objects between organizations at different level of granularity, and supports access control (AC) in heterogeneous environments. In LBAC, we introduce a new concept – AC layer, which wraps the AC systems of a virtual organization. The security information exchanges between the AC systems inside and outside the AC layer through a well-defined interface of the AC layer. The AC layer seamlessly glues different AC systems in organizations together according to the inter-organization workflow. Inside an organization the role is used to manage accesses. The encapsulation and dynamic binding features of the layered based approach make LBAC suitable for the collaborative design and manufacturing environment.

Keywords. Access control, virtual enterprise, collaboration, task assignment, security.

1 Introduction

Design and manufacturing of complex products requires collaboration among many companies. Security issues and access control in collaboration have attracted attention recently [2][15][19][3][20][30]. To share information and resources among untrustworthy users is the motivation of designing secure systems. Security is a broad concept that covers a lot of different areas (from encryption to access control). In this paper, we only discuss one aspect of security – access control.

Generally speaking, Access Control (AC) deals with how security systems control users' actions performed on environmental information and resources

[6][9]. In collaborative design and manufacturing, the AC system should facilitate sharing information among organizations; yet protect intellectual properties and trade secrets. There are three basic functions of an AC system: (1) managing users who are allowed to access resources; (2) protecting resources existing in the environment; (3) controlling actions that a legal user performs on a right resource.

Several AC models have been introduced such as the Discretionary Access Control (DAC) [31], Mandatory Access Control (MAC) [5], and Role-based Access Control (RBAC) [31]. Recently, to increase flexibility and capability, the AC models have been extended. Thomas [33] controls access based on tasks instead of individual objects. Kang [20] and Zhou [35] split one AC system into two parts (internal and inter-organizational) to support collaboration within and between organizations. Argus-Systems[1] and Lupu [25] decompose an entire AC system into hierarchical levels, and in each level different modules are implemented, these modules in different levels can be easily integrated to enforce different practical AC policies. Finding cross-platform role to support control information sharing between different organizations [30]. Several AC systems [6][8][23][24][32] have been developed for virtual enterprise systems. And also some AC research [34][17][22] integrates AC with workflows.

In this paper, we present an access control of workflow in a collaborative environment, called Layer-based Access Control (LBAC). LBAC supports AC dynamically, protects objects in different granularity, supports heterogeneous collaborative environments, and provides a uniform way to handle internal and inter-organizational AC. This paper is organized as follows, in section 2 we briefly introduce the background of existing AC models and workflow. In section 3 we address functional requirement of AC in a Design and Manufacturing Environment. In section 4, we introduce LBAC. Section5 explains how the framework exchange access control information and dynamically adjust their owner security policies in accordance with the requirements of the entire process in which they collaborate. In section 6 an example is given to illustrate how our framework works. Finally we give our conclusion.

2 Background

2.1 Existing Access Control Models

AC models are used to define and handle the relationships among three basic entities (“who”, “which” and “what”) [12]. “Who” refers to the subject, “which” represents the object, and “what” is the action that a subject performs on an object. The AC is enforced by describing the access *relationship* among subjects, objects and actions. Access Matrix (AM) [31] is a straightforward way to describe this relationship. But in real implementation, Access Control Lists (ACLs) or Capability Lists (CLs) are usually employed instead of using AM [31]. AM, ACLs and CLs are low-level AC mechanisms used to directly control access. How to generate, update and maintain these mechanisms is solved by high-level models, such as DAC, MAC, RBAC, TBAC etc. These models convert real AC requirements into policies and enforce these policies into an AC system by using those low-level mechanisms.

Several existing AC models have been used to create and update low-level AC mechanisms. MAC [5] controls the access based on the classification of subjects and objects in the system. Write up and read down principles are held to guarantee that information flows from the lower level to the higher level. In MAC, once the system is implemented, the AC is hard to be modified. DAC [31] adopts owner-based administration of access rights, which makes the system more flexible. The owner of an object, who is usually its creator, has discretionary authority over who else can access that object. In DAC, the control of the information flow is limited. In RBAC [31], the AC is separated into two phases: (1) associating users with roles and (2) controlling access of roles. Role is the base of RBAC. Role is defined as "a set of actions and responsibilities associated with a particular working activity." [31] This three AC models handle access from the subject-object viewpoint. These approaches are designed separately and use different ways to control access. All of them need centralized subject and object information, and they are applied on organizations with fixed boundaries.

AC models have been extended to solve problems in distributed environments. Task-based Authorization Controls (TBAC) [33] has been proposed. TBAC is a task-oriented model for AC and authorization. The core of TBAC is the active security model that is well suited for distributed computing and information processing activities with multiple points of access, control, and decision-making. TBAC focuses on security modeling and enforcement from the application and enterpriser perspective rather than from a system-centric subject-object view. In the TBAC paradigm, permissions are checked-in and checked-out in a just-in-time fashion based on activities or tasks.

The SALSA Security Architecture [20] is another extension, which handles AC within and among organizations. These organizations are connected by inter-organizational workflow. There are two kinds of AC modules in the overall security architecture: (1) an organization-specific access control module (OACM) that is controlled by each organization and enforces a security policy that is set by each organization; (2) a task-specific access control module (TACM) that is controlled by each workflow and enforces task-specific security policy. Only a person with intimate knowledge of the workflow can set the security policy of each task because a task-specific security policy depends on the semantics of the workflow.

W-RBAC [34] is an AC model designed for workflow system and supports both static and dynamic (history based) constraints. In this model the permission system is defined separately from workflows. The permission system works as an organization model, in which users, roles, privileges and business constraints are defined. "The permission system answers the workflow query and orders the users that satisfy the query according to some criteria." [34]

SecureFlow [17] specifies and enforces complex security policies within a workflow. And each task in the workflow associates with Authorization Templates (ATs) that are stated by the workflow designer. According to ATs authorization is actually granted when the task is started, and authorization is derived from a centralized AC database.

2.2 Collaborative Environment for Design and Manufacturing

In design and manufacturing environments, a process can be decomposed into a hierarchical level structure, where each level consists of subtasks. Several different approaches were proposed to represent this hierarchical decomposition of processes, such as and/or graph[18], Bayesian Network[29], hierarchical Petri Net[10][11] and process grammar[4][7]. The hierarchical decomposition of a process can be easily used to support the division of labor, because a process usually is decomposed according to its functions/requirements. Each subtask represents relatively independent functions/requirements in the process, so the subtasks can be handled by different organizations based on the division of labor. The hierarchical decomposition makes design and manufacturing collaboration special.

A company may outsource part of its work (i.e., subtasks of the process) to other companies. In this paper, task outsourcing is called task assignment, which brings different organizations to form a virtual enterprise[14][15][16][26][28]. Tasks are assigned dynamically based on the running status of a process, and task assignment decides an organizations joining or leaving a virtual enterprise and also defines the relationship between them. Task assignment also requests AC information to be sent from the company to its subcontractors, and this kind of security requirements haven't been addressed by other AC models for Virtual Enterprises and workflows. In our research we integrate AC requirements into workflows that are described by using process grammar.

Process Grammar

Our AC model uses the terms of process grammar for process representation and task decomposition. Process grammar [4] has been proposed to represent design and manufacturing process and to generate process flow dynamically. Process flow graphs describe the information flow of a design methodology, and process grammars provide the means for transforming high-level task into progressively a more detailed set of tasks as well as selecting a method among many alternatives for a task.

In the process grammar, the process flow graph consists of two types of entities: *tasks* and *data specifications.* A task is a single unit of design activity as defined with the process flow diagram. The flow diagram shows how to compose a task and the input and output specifications of the task. Data specifications are design data, where the output specification produced by a task can be consumed by another task as an input specification. There are two types of tasks, a *logical task* and an *atomic task*. A logical task can be decomposed into a set of subtasks. An atomic task is the simplest form of the task, which cannot be decomposed any further. Invoking an atomic task represents the execution of an application program or a specific tool. In essence, an atomic task is defined as

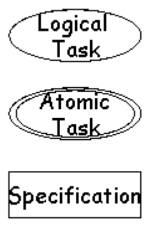


Fig 1. Three elements

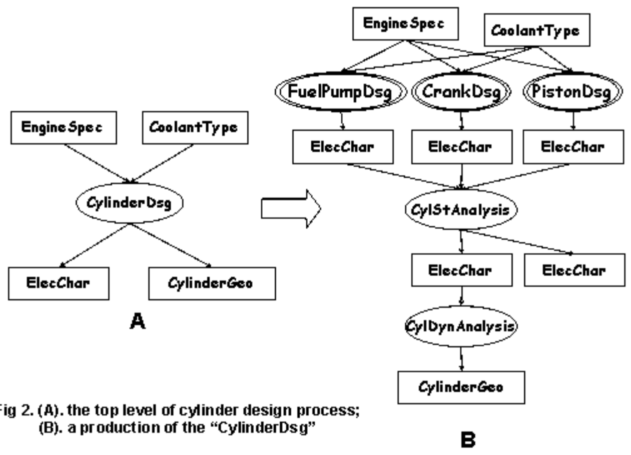


Fig 2. (A). the top level of cylinder design process;
(B). a production of the "CylinderDsg"

an encapsulated tool. The three basic elements are shown in Fig 1. Fig 2 shows an example (A) top level process flow of "cylinder design" and (B) a production of logical task "CylinderDsg". These flow graphs allow designers to visualize process alternatives.

The process grammar provides an abstraction mechanism so that designers are not overly burdened with details. Process grammar allows a user to represent and manipulate a small number of abstract, higher-level tasks that can be expanded into detailed, executable alternatives. This can be especially valuable when engineers from different disciplines are working together on a project. Process grammar can also support the dynamic, iterative nature of the design process. During the execution of a process, if the execution of certain task does not meet the requirement, a roll back can occur to an appropriate point and a new production can be applied to generate alternative process flow dynamically.

3 Functional Requirements of Access Control in a Design and Manufacturing Environment

The main characteristics that distinguish the AC systems of design and manufacturing from other general AC systems are the hierarchical process decomposition and task assignment. As explained in section 2.2, there are two types of tasks: *logical task* and *atomic task*. *Task assignment* refers to an entire procedure from choosing subtasks, giving them to other companies, to receiving the final results of these subtasks. The assigned tasks are called *assignments*. An *assigner* is the one who gives an assignment, and an *assignee* is the one who receives and handles the assignment. After an assignment is completed, the assignee returns the results to the assigner. Sharing distributed objects among distributed subjects challenges the designing of an AC for a virtual enterprise. In a virtual enterprise, protecting processes/workflows themselves is as important as protecting the data. Sometimes, it is necessary to protect the data and sub-processes of a task from the assigner who request the task to do. The assigner of a task may not have the right to view all the details of sub-processes, or data generated inside of the task. Yet sometimes the assignee of a task is required to follow a guideline of processes and show all the intermediate results and the workflow to the assigner.

An AC system for a design and manufacturing needs to support following functions:

1. Protect distributed objects within an organization and between organizations (inter-organization). Accesses of objects come from two different directions: (1) from the subjects who are in the same organization with the objects, (2) from the subjects who are in other organization. The trustworthiness of those subjects is different, so the protection should be different.
2. Protect objects in different levels. Based on process hierarchical decomposition, objects are divided into three different levels: logical task, atomic task and individual object (such as a car blueprint, a CAD tool). An AC system in a virtual enterprise should protect not only individual objects but also objects in different granularities. It should also provide a coherent AC setting view of logical tasks, atomic tasks and individual objects.
3. Support dynamic organization relations. Task assigning dynamically connects organizations together, dynamic inter-organizational protection is needed. An AC system should provide a consistent way to handle both internal and inter-organizational protections.
4. Support heterogeneous collaborative environments. Each organization has its own AC policy and AC system. A virtual enterprise AC system should support different collaborative environments.
5. Support different level of collaborative environments. Based on division of labor, the collaboration is not limited in organization level. Task assignment occurs in the group level within an organization as well as in the level above virtual enterprises, so AC systems need to support collaboration in different level.

The classical AC models, such as MAC, DAC and RBAC, are not suitable to solve the AC requirements in virtual enterprise [15][13][19]. First, a virtual enterprise subjects and objects are distributed in different organizations that dynamically join or leave, and no centralized subject and object information is available, so directly using these models to create one AC system for a virtual enterprise is difficult. Moreover, the classical AC models are designed for organization with fixed boundary, but a virtual enterprise does not have a fixed boundary. The classical AC models are designed for AC inside an organization, so they do not support inter-organizational AC information exchange.

TBAC [33] and SALSA Security Architecture [20] do not suit well a virtual enterprise for design and manufacturing because of the assumption they make. They assume that all details of an entire process are clearly defined before the process is started, which is not true for a design and manufacturing process. In such environment, there are several different ways to implement a logical task. The actual implementation usually is selected according to the running status of the process, and the detail object information of the process cannot be defined when the process is designed, so TBAC and SALSA Security Architecture do not suit a dynamically changed virtual enterprise.

For an entire virtual enterprise an AC system can be centralized or be distributed. To create one centralized AC is difficult because it is difficult to find someone who can be trusted by all members in a virtual enterprise to manage the AC system. W-RBAC [34] and SecureFlow [17] are designed using a centralized AC system, which suit the security requirements of a single company but it is hard to extend them to fit dynamically changed virtual enterprises. In a distributed AC system, it is necessary to have a standard interface through which the original AC systems in organizations can talk with each other to form a distributed AC system for the entire virtual enterprise.

4 Layer-Based Access Control Model (LBAC)

In this section we present a Layer-based Access Control (LBAC) model that supports distributed AC systems for a virtual enterprise, and meets the requirements of distributed design and manufacturing. LBAC protects objects between organizations at different level of granularity, and support AC in heterogeneous environments. In LBAC, an AC layer is a wrapper that contains the original AC system of an organization unit, and it provides an interface through which the original AC system can talk with outside. LBAC uses role based access control within a unit. LBAC extends AC across the AC layers, and provides support for organization unit management, role-organization mapping and AC information exchange among different organization units.

4.1 Three Entities

In LBAC, the three access control entities, namely, *subject*, *object* and *action* [12], are defined as follows.

Subjects have two types: *users* and *organization units*. A user is modeled by his/her username, and captures some other information, such as skills. An organization unit is the structure of the organization. A unit may represent a small group inside a company or an entire virtual enterprise. A unit is identified by its name and also captures its functionality. In LBAC a *role* is a set of *responsibilities* that are allowed to be performed by users who act the role. The AC in LBAC is also separated into two steps: (1) associating users with role, (2) controlling role's actions on objects. When a user associates with a role, the user's skills are checked. Only when the user's skills satisfy the responsibilities of a role, the user can act the role. Roles are organized into organization units.

Objects correspond to the information to be protected. In LBAC, such information includes not only the design data but also tasks and the process flow.

Objects have three types: (1) individual objects, (2) a collection of individual objects associated with an atomic task, (3) a collection of individual objects associated with a logical task. We mainly focus on how to control actions performed on the first two type objects, namely, terminal tasks and logical tasks.

Different objects are associated with different *actions*. For example , a car design blueprint can be viewed, revised, deleted and even copied. Those actions are tightly related to each real object. In this paper, we rather focus on actions requested by process hierarchical decomposition and task assignment. They are "view", "execute" and "assign". Other actions can be added according to the practice requirements.

– "View" action is an atomic task that means reading the information about the task, and also reading the information about the individual objects related to the task. Since a logical task can be decomposed into more than one level, "view" permits roles get the first level detail of the logical task decomposition.
– "Execute" action is applied on logical tasks only. Because a logical task usually has several alternative implementations, "execute" lets roles decide which implementation is chosen for the logical task in the process runtime.
– "Assign" action also is applied to logical tasks only. "Assign" action is required by task assignment, which allows roles to select an assignee and to outsource the logical task.

4.2 LBAC Functionalities

Subject Management

The role concept of RBAC [12][31],in LBAC, is adopted to handle the AC inside an organization. In LBAC, the subject management contains three parts: user management, role management and user-role mapping. User management and user-role mapping are handled in the real work after the system is shipped. In LBAC, roles are group-oriented, and roles are defined within a group. There are two types of roles: manager and common roles. The manager is a built-in role who has responsibilities to manage his group, such as creating, modifying and deleting roles in his group, and associating users with roles in his group, and so on. Common roles are created during the real work based on the real requirements, and the responsibilities of these roles are specified by the manager.

Organization Unit Management

A group, which is a collection of manager and common roles, is a basic organization unit in LBAC. By associating roles with other groups, groups automatically support hierarchical enterprise structure. A group is managed by the administrator of the organization. The administrator specifies the group name, functions and the manager, and associates the manager with a user, and then the group becomes alive and the manger starts to manage his own group. In addition granting role to users, the manager can grant a role to other groups.

Task Assignment and Basic Policies for Permission Setting

Different permission setting policies are enforced depending on the execution modes of processes. There are two execution modes: *Single Execution Mode (SEM)* and *Collaborative Execution Mode (CEM)*. In SEM, the entire process is totally handled by one role, and only one collaborative action, "view", can be performed here. The basic AC policy used in SEM is DAC, namely, the owner (who creates the object) decides which role can view his object. The roles must in the same group with the owner.

In CEM, the process requires more than one role to finish the work. There are two sub-modes in the CEM: (1) *task assigning within one organization*, (2) *task assigning between organization units*. Based on different trustworthiness, the two sub-modes need different policies.

Task assigning within an organization: When a task is assigned within an organization, the assignment is called *job*, which is given from one role to another within an organization. A job contains two parts: a job template (containing all necessary information for the assignee to complete the job) and a contract (exchanging AC information). After the assignee accepts the job, there are two different AC requirements: (1) access to the job-related objects is totally controlled by the assignee. In this case, the job is executed in SEM and all access is controlled by the assignee; (2) access to the job-related objects is controlled by both the assigner and the assignee. A concept "*contract" is* introduced to let the assigner control the access to those objects. A contract defines a set of permissions that are applied to the entire job. The AC system automatically inserts the permission settings into ACL of each job-related object when the object is created. Through the contract the assigner can control access to objects. The contract is generated by the assigner, but it is a negotiated result between the assigner and the assignee. Other part of AC system is the same as the one used in SEM. In this mode, the ACLs update information comes from two parts: from the object owner and from the contract.

Task assigning between organization units: The assignment between organizations is called *project*. A project also contains two parts: a *project template* and *contracts*. Similar to a job template, a project template contains all necessary information for assignee organization to complete the project. Different from job, a project has two contracts: a *report contract* and a *local contract*. The report contract defines what object information should be returned from the assignee to the assigner. This information will be explained by the AC system in the assignee side. The local contract defines who (roles in the assigner's organization) can access the objects information returned by the assignee. The local contract is explained by the assigner's AC system. The project template and the report contract are sent to the assignee organization, and the local contract is kept in the assigner's system.

From the viewpoint of AC, the object information of a job (an assignment within organization unit) is different from the one of a project (an assignment between units). When a job is done, we assume that all job-related objects are available to all roles in the same organization. So using a contract to exchange

AC information is enough. But a project is assigned across the boundary of organizations, and the project-related objects belong to the assignee's organization. For security reasons, these objects should not be directly accessed from outside of the organization. A *report* is designed to solve this problem. According to the report contract, the assignee's AC system puts the requested object information into the report and returns it to the assigner organization. Based on the local contract, the assigner AC system sets permissions for each objects reported by assignee. The assigner side ACLs are updated by the AC system according to the local contract; the assignee side ACLs are updated by the object owner and AC system based on the report contract.

Inheritance of Access Control

To handle object in different levels, an inheritance policy is enforced in LBAC. Inheritance policy requests that the permission settings of higher-level objects (such as logical tasks and atomic tasks) will be inherited by lower-level objects (e.g. atomic tasks and individual objects). Here we use a simple example to illustrate permission inheritance. Suppose manager has "view" permission of logical task "T1", he will be automatically granted "view" permission of all objects (e.g. other logical tasks or atomic tasks) related to "T1" if the object supports "view" permission. Similar to logical tasks, the permission settings of an atomic task will be inherited by all individual objects belonging to the atomic task. Even through permission setting can be inherited automatically, but inheritance can be terminated by changing the setting of sub-object. Continue the previous example, if inside "T1" there is another sub logical task "T2", according to the automatic inheritance, manager can "view" details of another level of "T2". Repeating those steps, the manager can "view" all detail of "T1" until all individual objects while "T1" is fully worked out. But we can terminate this automatic inheritance by removing the "view" permission of manager from "T2". After removing, the manager can view all the details of "T1" except the objects related to "T2" after "T1" is fully worked out.

AC Information Exchange

In LBAC model, an interface (a contract and a report) is used to (i) exchange AC and object information; (ii) to hide the AC system detail in an organization from outside; (iii) to provide a stand way to extend the original AC systems to let them talk with each other; (iv) to convert inter-organizational AC into internal AC. The interface provides mechanism that brings a contract and inter-organizational AC requirements into the organization's AC system. Through the report, outside subjects are allowed to access inside objects, and the internal AC system is extended to handle inter-organizational AC by using the internal way. The same approach can be used in different collaborative environments from group level to virtual enterprise level.

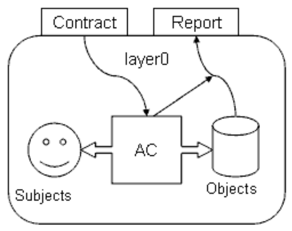


Fig 3. Basic structure of AL

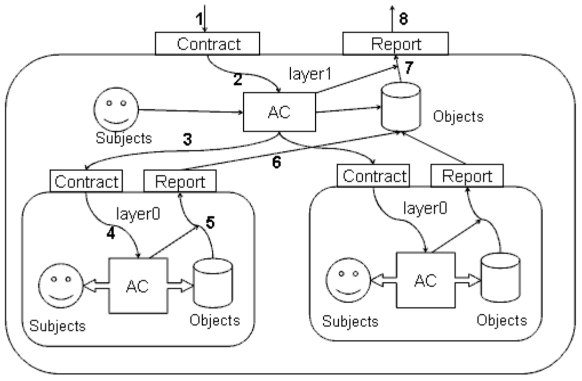


Fig 4. The structure of multiple-layer

4.3 Architecture of LBAC

A Virtual Enterprise dynamically bring different organizations together and also bring different AC systems together to form an integrated AC system for the entire virtual enterprise. The major consideration of designing LBAC is to allow different AC systems to freely exchange AC information with each other so that those different AC systems can be integrated easily. To archive this goal, the Object-Oriented (OO) approach is used to support several OO features: abstraction, encapsulation, messages and dynamic binding. These features are very suitable for developing an AC system. In LBAC, an entire AC system is treated as an object. To differentiate with the concept "object" used inside the AC system, the entire AC system (including all subjects, objects and actions) is called as an access layer (AL). An AL is an abstract of an entire AC system. The AL wraps the whole AC system inside.

All information (subjects, objects, actions and AC information) is hidden by the AL. The information is exchanged by using messages through the interface of the AL. An interface contains two parts: a contract and a report. The contract is used to receive AC information from outside, and the report is used to send process-related object information to outside. This is same as the general object in OO approach; the interface is designed for an AL to allow the functions inside the AL to exchange messages with outside. Because the interface is used, the AL supports dynamic binding, which occurs when a task is assigned. The basic AL is shown in Fig 3.

Fig 4 shows the architecture of an AC system with multiple-AL, and the steps of how the information is exchanged between the adjacent ALs. (1) Layer1 receives AC information through its contract form outside higher AL; (2) the information is merged into its own AC system. When a task is assigned from layer1 to layer0, (3) the AC system in layer1 creates a contract and sends the contract to layer0, and (4) AC information in the contract of layer0 is also added into layer0's AC system, so that the AC information is sent from a higher AL to a lower AL following the task assignment. But the object information is different.

In accordance with the AC system inside an AL, certain objects are contained in its report, and these objects can be accessed from outside. (5) The objects in a report of a lower AL (e.g. layer0) are (6) added into the object set of the higher AL (e.g. layer1). As illustrated in Fig 4, the object set in layer1 contains the objects in the reports of layer0. These objects could be part of the report of layer1. Then go through (7) and (8) object information (data or process) is sent to higher layers. Another feature of LBAC is that the higher AL always hides information of the lower ALs inside it. So the entire AC system for a virtual enterprise is like an onion, the outer AL encapsulates the inner ALs. By using LBAC the difficulties of the AC system for a collaborative design and manufacturing environment are solved.

The OO features make LBAC suitable for AC of a collaborative environment. Because of encapsulation, each organization can keep its own AC system. Interface and message provide a standard way to allow any different AC systems communicate, so that the original AC system in each organization is extended to handle internal AC and inter-organizational AC at the same time. Dynamic binding supports dynamic connection between any AC systems in different organizations according to the task assignment. By treating logical tasks and atomic tasks as objects, protecting objects in different granularities are archived: the permission settings of a logical task is sent from higher layer to lower layer through contract, and the object information is sent from the lower layer to the higher layer through the report according to the AC requirements. By merging AC information received from contract and sending out objects through report, LBAC create a uniform way to handle AC internal to an organization and inter-organization. By using LBAC, a distributed AC system can be easily formed for a dynamically changing virtual enterprise.

4.4 Comparison with Other AC Models

LBAC is an extension of the classical AC models. Compared to TBAC, LBAC does not request all process detail at the starting time. LBAC uses the same way to protect logical objects and individual objects existing in the system according to the status of the process. The more details of a process are created, the more objects LBAC system is protected. Different from SALSA Security Architecture, LBAC integers the internal AC system and the inter-organizational AC systems together, and LBAC reduces the complication of AC system and simplifies the management of the AC system by providing a uniform view to handle internal and inter-organizational AC. Another difference of AC design between SALSA and LBAC is the time when the access permissions are set. In SALSA the permissions to an object are decided at the design-time of the process. So during runtime, the permission settings cannot be changed. On the contrary, the permissions in LBAC are set during runtime, and can be dynamically changed when the process is executing. The third difference is that the AC in SALSA focuses on controlling the access initiated by a task to information (data). The LBAC deals with the access between system roles and tasks running in the system.

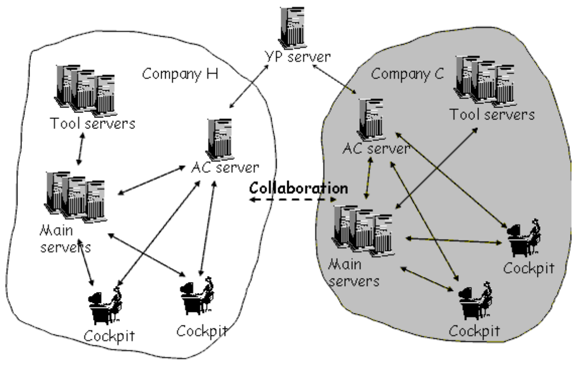


Fig 5. Structure of MIDAS

5 Implementation

MIDAS is process grammar based collaborative design and manufacturing environment. However, our approach is not only suitable for MIDAS, but also supports any collaborative environment where tasks are hierarchically decomposed and assigned dynamically.

MIDAS Architecture

Fig 5 shows the structure of MIDAS. Each company has its own process management system, which can be either a MIDAS system, or a web service system interfaced with MIDAS. A MIDAS system consists of Main Servers, Tool Servers, AC server and cockpits.

Main servers are the process execution engines, which handle the process management including creating, managing, and maintaining process information libraries, executing processes, invoking external tools and storing the results. *Tool servers* actually invoke the external tools (such as the existing CAD/CAM systems, 2D/3D simulation systems and even human beings) after receiving invoking request from Main Servers. *AC server* is the central part of access control subsystem and each company only has one AC server. The main functions of AC (Access Control) server are user management, group management, authentication, access request monitoring, granting and denying, policy maintenance etc. *Cockpits* are client side programs, through which users interact with MIDAS. By using cockpit, the users can create, execute process, and manage process information libraries. Administrator can manage user, group, and member in each group. Group manager can maintain his own group. *YP server* is the information center among different MIDAS systems. It provides necessary information to allow servers in different MIDAS systems to communicate with each other.

All servers are developed using Java. Communication between servers or between servers and cockpits are using Java RMI technology. Because of the portability of Java technique, the servers can be run on different platforms.

```
<Specification objectName="EngineSpec">
    <own memberName="Manager"/>
</ Specification>
<logic objectName="CylinderDsg">
    <own memberName="Manager"/>
    <view memberName="CrankEgr"/>
    <execute memberName="Manager"/>
</logic>
...
```

List 1. Part of ACLs

Cockpit is implemented as a Java applet, so that it can be involved in any Java enabled web-browsers.

Data Representation in MIDAS

All AC related data, such as ACLs (list 1), user information, group information and contract are represented by using XML, which makes this data easy to understand and managed. From this point of view, our AC system is an XML-based AC system. Similar to our purpose, IBM introduces XACL (XML Access Control Language) [21], which enables AC system designer to specify and execute fine-grained and complex authorization policies. Comparing with XACL, the main purpose of using XML is different. Our approach uses XML to record AC information; XACL uses XML to describe different AC models based on standard XML schema [22].

6 Example

In this example, we illustrate how a complex process is handled by collaboration of different companies and how access is controlled. Suppose that company H plans to design a new engine, and starts a new project called "engine design". The engine design process is decomposed into different subtasks. We further assume that some subtasks are handled by other engineering companies. Fig 6 shows the top-level process flow of the project. The company H decides to outsource a part of the process. Company C is a professional cylinder design company. Company H asks company C to design the cylinder part for the new engine. The detail AC of assigning the logical task "CylinderDsg" as project "Cylinder Design" from company H to company C will be introduced later.

The "Cylinder Design" is a simpler subprocess. One rule used here is "only the manager of a company can accept and starts a project assigned from outside". Before the project is started, no project-related objects exist in MIDAS.

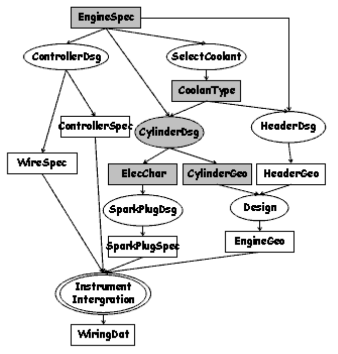


Fig 6. The top level process flow of "engine design" project

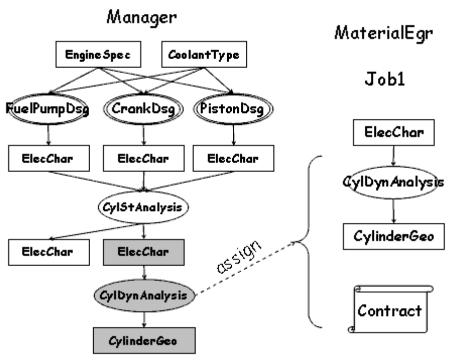


Fig 7. Task assigning between users in the same group

After that, the manager of company C starts the project "Cylinder Design" (fig 2 (A)). According to the process flow, five objects (data "EngineSpec", data "CoolantType", logical task "CylinderDsg", data "ElecChar" and data "CylinderGeo") are created by Main server. For the detail of process management, please read [27]. Here we illustrate AC part only. AC system creates ACL for each object (List 1).

Because the "Cylinder Design" process is simple, it can be handled by the manager of company C. So MIDAS is working in SEM. As owner of all objects, manager control all access to these object. In List 1, "manager" grants view permission of logical task "CylinderDsg" to "CrankEgr" and execute permission to himself. Based on the ACLs the "CrankEgr" can view the first level detail of "CylinderDsg" and "manager" can execute it. After manager executes the "CylinderDsg" the entire process is extended as shown in Fig 2 (B). Several new objects are created and new ACLs are also created. The manager can repeat the steps to set permission for each object and execute logical tasks until the entire "Cylinder Design" process is done. Then return the result to the company H.

While logical tasks are executed, the process becomes more elaborated. Maybe the manager needs help. So he will assign tasks to others, then the MIDAS execution mode shifts from SEM to CEM. Fig 7 shows how a task is assigned within an organization unit. First the manager selects a task and an assignee (e.g. "MaterialEgr"), and then creates a job "Cylinder Dynamic Analysis". The job contains two parts: a job template and a contract (fig 7). List 3 shows a simple job contract. According to this contract the "Manager" gives himself "view" permission for the entire job. By using this contract, the AC system gives manager "view" permission of all job-related objects. Through the contract the assigner can control the access to the objects. The other part of the AC system is same as the one used in SEM. The "MaterialEgr" can decide permission settings of all job-related objects.

The second case in CEM is task assignment between organizations. Figure 6 illustrates how company H assigns "Cylinder Design" subprocess to company

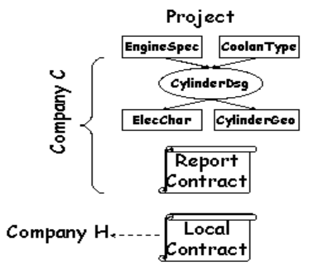


Fig 8. A project assignment

C. The manager of Company H selects logical task "CylinderDsg" and generates project "Cylinder Design" for Company C. The project contains two parts: a project template and contracts (Fig 8). The project has two contracts: report contract and local contract. The project template and report contract are sent to the assignee organization, and local contract is kept in the assigner's system. During the project execution, the assignee's AC system puts the requested object information into the report and returns the report to the assigner organization based on the report contract. After the assigner receives the report, the assigner's AC system automatically creates ACLs for those objects in the report and set permissions based on the local contract.

7 Conclusion

In this paper, we discussed the special access control requirements raised in a collaborative environment for design and manufacturing. We have identified the functional requirements of AC system, where tasks are decomposed hierarchically, and assigned to groups within/outside of organizations. Our proposed layer based access control extends AC across the AC layers, and provides support for organization unit management, role-organization mapping, and AC information exchange among different organization units in a virtual enterprise. We have shown that LBAC supports AC in heterogeneous environments, and provides a mechanism that protects objects between organizations at different levels of granularity. Currently, a prototype of the layer-based access control model is being implemented using JAVA to make our system platform-independent.

References

[1] Argus-Systems. Pitbull lx secure application environment (white paper). 2001. 198

[2] V. Atluri and W. Huang. An authorization model for workflows. In *Proceedings of the Fourth European Symposium on Research in Computer Security*, pages 25–27, Rome, Italy, September 1996. 197

[3] R. Au, M. Looi, and P. Ashley. Automated cross-organisational trust establishment on extranets. In *Australian Computer Science Communications, Proceedings of the workshop on Information technology for virtual enterprises*, volume 23, 2001. 197

[4] R. Baldwin and M. J. Chung. Design methodology management: A formal approach. *Computer*, 28(2):54–63, 1995. 200

[5] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations and model. *TR. M.*, pages 74–224, 1973. 198, 199

[6] A. Bullock and S. Benford. An access control framework for multi-user collaborative environments. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, pages 140–149, Phoenix, Arizona, United States, November 14-17 1999. 198

[7] M. Chung, P. Kwon, and B. Pentland. Making process visible: A grammartical approach to managing design processes. *ASME Transaction, Journal of Mechanical Design*, 124:364–374, 2002. 200

[8] G. Coulouris, J. Dollimore, and M. Roberts. Role and task-based access control in the perdis groupware platform. In *3rd ACM workshop on Role-based Access*, pages 115–121, Fairfax, VA, 1998. 198

[9] P. Dewan and H. H. Shen. Flexible meta access-control for collaborative applications. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 247–256, Seattle, Washington, United States, November 14-18 1998. 198

[10] W. Emmerich and V. Gruhn. Funsoft nets: A petri-net based software process modeling language. In *Proc. of the 6th Int. Workshop on Software Specification and Design*, pages 175–184, Como, Italy, 1991. IEEE Computer Society Press. 200

[11] S. Erdmann and J. Wortmann. Enterprise modelling with funsoft nets. In *Proceedings of the 1st International Enterprise Distributed Object Computing Workshop (EDOC'97)*, Gold Coast, Australia, October 24-26 1997. 200

[12] D. F. Ferraiolo and et al. J. Cugini. Role based access control: Features and motivations. In *Computer Security Applications Conference*, 1995. 198, 203, 204

[13] R. T. Fielding, E. J. Whitehead, K. M. Anderson, A. G. Bolcer, P. Oreizy, and R. N. Taylor. Web-based development of complex information products. *Communications of the ACM*, 41(8):84–92, 1998. 202

[14] M. S. Fox and M. Gruninger. Enterprise modelling. *AI Magazine*, pages 109–121, Fall 1998. 200

[15] M. Hardwick, D. L. Spooner, T. Rando, and Morris K. C. Sharing manufacturing information in virtual enterprises. *Communications of the ACM*, 39(2):46–54, 1996. 197, 200, 202

[16] Y. Hoffner, H. Ludwig, P. Grefen, and K. Aberer. Crossflow: Integrating workflow management and electronic commerce. 200

[17] W. Huang and V. Atluri. Secureflow: A secure web-enabled workflow management system. *ACM Workshop on Role-based Access Control*, pages 83 – 94, 1999. 198, 199, 203

[18] IDEF. http://www.idef.com, 1998. 200

[19] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(2), 2001. 197, 202

[20] M. H. Kang, J. S. Park, and et al. Access control mechanisms for inter-organizational workflow. In *In Sixth ACM Symposium on Access Control Models and Technologies*, 2001. 197, 198, 199, 202

[21] M. Kudo and S. Hada. Xml access control. http://www.trl.ibm.com/projects/xml/xacl/xmlac-proposal.html, 2000. 210

[22] M. Kudo and S. Hada. Xml document security based on provisional authorization. In *7th ACM Conference on Computer and Communication Security (CCS 2000)*, Nov. 2000. 198, 210

[23] P. Linington, Z. Milosevic, and K. Raymond. Policies in communities: Extending the odp enterprise viewpoint. In *Proc. 2nd IEEE Enterprise Distributed Object Computing Workshop*, San-Diego, 1998. 198

[24] E. Lupu, Z. Milosevic, and et al. Use of roles and policies for specifying, and managing a virtual enterprise. In *the 9th IEEE International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE'99*, Sydney, Australia, 1999. 198

[25] E. Lupu and M. Sloman. A policy based role object model. In *In Proc. 1st IEEE Enterprise Distributed Object Computing Workshop*, pages 36–47, Gold Coast, Australia, 1997. 198

[26] NIIIP. The niiip reference architecture. www.niiip.org, 1996. 200

[27] Y. Qin. *Manufacturing Infrastructure and Design Automation System (MIDAS) with XML representation*. PhD thesis, Computer Science and Engineering, Michigan State University, East Lansing, 2002. 211

[28] I. Rahwan, R. Kowalczyk, and et al. *Virtual Enterprise Design - BDI Agents vs. Objects*. Recent Advances in Artificial Intelligence in e-Commerce. R. a. L. Kowalczyk, M, Springer-Verlag, 2000. 200

[29] M. A. Rodrigues, Y. Liu, L. Bottaci, and D. I. Rigas. Learning and diagnosis in manufacturing processes through an executable bayesian network. In *13th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE-2000*, pages 390–395, New Orleans, June 19-22 2000. 200

[30] H. Roeckle, G. Schimpf, and R. Weidinger. Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. *ACM*, pages 103–110, 2000. 197, 198

[31] R. S. Sandhu and P. Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, pages 40–48, September 1994. 198, 199, 204

[32] G. Stevens and V. Wulf. A new dimension in access control: studying maintenance engineering across organizational boundaries. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, New Orleans, Louisiana, USA, November 16-20 2002. 198

[33] R. K. Thomas and R. S. Sandhu. Task-based authorization controls (tbac): A family of models for active and enterprise-oriented authorization management. In *the IFIO WG11.3 Workshop on Database Security*, Lake Tahoe, California, 1997. 198, 199, 202

[34] J. Wainer, P. Barthelmess, and A. Kumar. W-rbac - a workflow security incorporating controlled overriding of constraints. *International Journal of Cooperative Information Systems*, 12(4):455–485, 2003. 198, 199, 203

[35] Q. Zhou and C. B. Besant. Information management in production planning for a virtual enterprise. *Int. J. Prod. Res.*, 37(1):207–218, 1999. 198

Secure Double Auction Protocols with Full Privacy Protection

Changjie Wang[1], Ho-fung Leung[1], and Yumin Wang[2]

[1] Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Hong Kong, China
{cjwang,lhf}@cse.cuhk.edu.hk
[2] State Key Laboratory of Integrated Services Networks,
Xidian University, Xi'an 710071, China
ymwang@xidian.edu.cn

Abstract. Many researches have been done on the strategies of double auctions, an important class of auction protocols that permit multiple buyers and sellers to trade simultaneously in a market. Some well designed *dominant-strategy incentive compatible* double auction protocols have been proposed. However, the security and the privacy issues in double auctions are seldom studied in the literatures. In this paper, we propose secure double auction protocols, which achieve full privacy protection of participants. That is, each bid/ask information is always kept secret, even when there is any collusion of participants. It is clear that our suggestion is stronger than other previous work in which assumptions that certain auctioneers never collude are made. To achieve full privacy protection, we employ homomorphic ElGamal encryption and distribute the private key among the all participants. In such a way, all participants jointly compute the outcome of the double auction without revealing any additional bid/ask information. Also, the proposed protocol is publicly verifiable, so that the robustness is assured. The communication and computation complexity of the proposed protocols are analyzed.

1 Introduction

With the rapid development of Electronic Commerce over Internet, auctions have become one of the most popular activities in electronic commerce during the past a few years. Besides the common single-sided auctions, such as English auction, Vickrey auction, *etc.*, double auctions (DAs) [1] are an important class of auction protocols that permits multiple buyers and sellers to trade in the market simultaneously. Since security and privacy issues have become crucial considerations in online auction systems, there has been a very fast-growing interest in designing secure auction protocols. In particular, Vickrey auctions, and the more general form of (M+1)st-price auctions, have attracted much attention. Many cryptographic auction mechanisms [2,3,4,5,6,7] have been proposed for securing auction systems.

For double auctions, however, most studies have been focusing on how to design *strategy-proof* DAs protocol [8,9,10] while relatively little research has been done on the privacy and security issues. Some *dominant-strategy incentive compatible* DA protocols have been proposed, in which the dominant strategy for each buyer/seller is to declare honestly his own true value [11]. While these protocols should be superior to others as less effect is needed for the participants to counterspeculate other participants' actions, it is more significant to keep the *bids* and *asks*[1] secret in these DAs as they are the true values of the auctioned item(s) as seen by the buyers and sellers. In this paper, we study the security and privacy issues in double auctions and propose two secure protocols based on the two existing *strategy-proof* double auctions protocols, respectively. We employ the homomorphic ElGamal encryption in our protocols, and the private key is distributed among all buyers/sellers, so that the full privacy protection and the public verifiability of protocols are achieved.

The remainder of this paper is structured as follows. Section 2 introduces the rules of double auctions, and presents the requirements for secure double auctions. In section 3, we propose in detailed two secure double auction protocols with full privacy protection. We perform an analysis on security and efficiency of the proposed protocols in Section 4, followed by conclusions in Section 5.

2 Double Auction

In this section, we introduce the rules of double auctions and present the security requirements for double auction market.

2.1 Double Auction Rules

Different from the single-sided auctions, in a double auction market, both the buyers and the sellers can bid to trade on the goods. Usually, a double auction includes two phases: bidding phase and opening phase. In the bidding phase, each buyer/seller submits his bid/ask to the auctioneer. In the opening phase, the auctioneer collects all the individual bids and asks to determine the total supply and demand in the auction market, calculates the transaction price, and matches buyers and sellers according to the total market information and certain predefined rules. The matching tactics of double auction can be carefully designed, such that incentive compatibility can be achieved.

2.2 Requirements for Secure Double Auction

In a *dominant-strategy incentive compatible* double auction protocol, it is a rational action for buyers and sellers to declare their true values of the goods in the bids and asks, as this strategy is the dominant strategy for them. Such information, however, is usually regarded as the commercial secret of the buyers and sellers, which is too sensitive to be made known to the public. Therefore,

[1] We shall follow the convention in the double auction literature, and use the term *bid* to refers to a buyer's declaration of value, and *ask* a seller's declaration of value.

it is very important to achieve the privacy protection in these double auction markets should they are to become popular. We divide the security properties required into two categories: *privacy* and *public verifiability.*

For easily illustration, we summarize the sensitive information in the double auction market as three kinds: (a)the bid/ask information; (b)the bid/ask statistics information(*i.e.*, the total supply and demand information of the market); (c)the corresponding information of bid/ask and the identity of buyers/sellers. In this paper, we introduce two concepts of privacy protection in double auctions, namely, *strong privacy* protection and *weak privacy* protection. *Strong privacy* protection means that information (a),(b),(c) are always kept secret during and after the auction, and the only information opened is the final results of the auction. Under *weak privacy* protection, only the information (a) and (c) are protected, while the total market information(*i.e.*information (b))[2] has to be open to the public in the opening phase. In later sections, we propose two secure double auction protocols that achieve *strong privacy* and *weak privacy* protection, respectively.

With the privacy protection requirements achieved, only the final results of the double auction should be known publicly. Consequently, it is necessary to achieve the *public verifiability* of outcome, that is, anyone should be able to verify the correctness and legitimacy of the results of the double auction.

3 Secure Double Auction Schemes

In this section, we propose two secure double auction schemes using the distributed version of ElGamal encryption algorithm and several zero-knowledge proofs like Schnorr's proof of knowledge of a discrete logarithm [12]and Chaum's proof of equality of discrete logarithms [13]. These proposed secure double auction protocols with full privacy protection are based on McAfee's protocol [8] and the protocol of Yokoo et al.[10] respectively, both of which have the property of being *incentive compatibility.*

3.1 Trust Assumption and Framework of Proposed DA Schemes

In most of the past work on secure auction protocols, the security of schemes relies on the assumption that the auctioneer(s) or other (semi-)trusted parties never participate in collusion of any form. Such an assumption, however, is not always a realistic one since it is always possible that the auctioneer(s) may collude to manipulate the auction.

In our proposed double auction protocols, therefore, we do not make any special trust assumption. That is, neither sellers nor buyers should trust any third-party, and it is also anticipated that some group(s) of sellers/buyers might collude with auctioneers. The key idea of our approach is to distribute the trust

[2] In some double auction protocols, say [8], the total market information has to be open so that the auctioneer can calculate the transaction price

and the computation on the sellers and buyers themselves. Since it is obviously impossible for *all* sellers, buyers and auctioneers to collude as a single group (otherwise the auction is unnecessary and protocols of cooperative resources allocation should be used), competition exists among the participants of the auction, and we only need to ensure that no sensitive information mentioned in section 2.2 can be retrieved unless all sellers/buyers share their knowledge with all others, which is impossible when the participants are rational. Full privacy protection is thus achieved.

The framework for our secure double auction schemes can be summarized step by step as follows. First, the auctioneer publicly announces the trade of a certain type of goods by publishing the goods' description, the registration deadline, the offer valuation range, and a description on how many possible offer valuations are distributed among the range. Then, those interested buyers/sellers publish their ID's on a open blackboard before the registration deadline. Finally, the participants jointly compute the winners (*i.e.*, the buyers/sellers who are eligible to trade) as well as the transaction price.

3.2 Distributed ElGamal Encryption

ElGamal encryption algorithm [16] is a probabilistic public key system that has the homomorphic property,[3] which is useful in our protocols.

The ElGamal encryption system can be extended to a distributed version as follows, which is applied in proposed secure double auction protocols. First, all participant jointly generate the keys of ElGamal encryption system in a distributed manner. That is, each participant chooses a arbitrary number s_i independently, then computes and publishes $h_i = g^{s_i} \bmod p$ along with a zero-knowledge proof of knowledge of h_i's discrete logarithm as in [12]. The public key of the system is then calculated as $H = \prod_i h_i \bmod p$, and the corresponding private key is $S = \sum_i s_i$. This is called ***distributed key generation***. To decrypt a ciphertext $\{x, y\} = \{g^a(\text{mod}p), H^a m(\text{mod}p)\}$, all participant need to jointly perform the ***distributed decryption***. That is, each participant computes and publishes $x_i = x^{s_i} \bmod p$ separately and proves its correctness as described in [13]. The plaintext can then be decrypted by computing

$$\frac{y}{\prod_i x_i} = \frac{H^a m}{g^{a(\sum_i s_i)}} = \frac{(\prod_i g^{s_i})^a m}{g^{a(\sum_i s_i)}} = m \bmod p.$$

[3] A public key system $E(m)$ is *homomorphic* if and only if $E(a)E(b) = E(ab)$, and it is *probabilistic* if and only if for any exact plain message, there may be many different encryption results by selecting different parameters.

3.3 A Secure Double Auction Protocol Based on McAfee's Protocol

In this section, we propose a secure double auction protocol based on the McAfee's protocol (called PMD thereafter)[8], which is one of the early *dominant-strategy incentive compatible* double auction protocols.

We first review the PMD protocol briefly. Let $b_1, b_2, \ldots, b_m$ be the m buyers' bids and $s_1, s_2, \ldots, s_n$ be the n sellers' asks. Define

$$b_{(1)} \geq b_{(2)} \geq \ldots \geq b_{(m)} \text{ and } s_{(1)} \leq s_{(2)} \leq \ldots \leq s_{(n)}$$

Note the reverse ordering for buyers and sellers: we use the notion (i) for the *i-th* highest bid of buyers, and the *i-th* lowest ask of sellers.[4] Choose k such that $b_{(k)} \geq s_{(k)}$ and $b_{(k+1)} < s_{(k+1)}$ hold. Clearly, for (1) to (k), the bid of the buyer is greater than the ask of the seller, so at most k trades are possible. Defining $p_0 = \frac{1}{2}(b_{(k+1)} + s_{(k+1)})$. If $p_0 \in [s_{(k)}, b_{(k)}]$, the buyers/sellers from (1) to (k) trade at price p_0. If $p_0 \notin [s_{(k)}, b_{(k)}]$, only buyers/seller from (1) to $(k-1)$ can trade: each buyer pays $b_{(k)}$, and each seller gets $s_{(k)}$. In the second case, since the bid of buyers $b_{(k)}$ is larger than the ask of sellers $s_{(k)}$, the amount $(k-1)(b_{(k)} - s_{(k)})$ is left over. It is usually assumed that the auctioneer receives this amount.

We now propose a secure double auction protocol with privacy protection, which is based on PMD protocol. Suppose there are m buyers and n sellers in a double auction market and there are K possible offering prices, denoted by $p_1, p_2, \ldots, p_K$, satisfy $p_1 < p_2 < \ldots < p_K$ and $p_i = p_1 + (i-1) \cdot \triangle price$ for $i \leq K$, where $\triangle price$ denotes the price step. All buyers and sellers first generate the public key of the ElGamal encryption algorithm in a distributed manner, as described in Section 3.2. That is, each buyer/seller arbitrarily chooses sb_j, $1 \leq j \leq m$ and ss_i, $1 \leq i \leq n$ respectively and publishes $h_j = g^{sb_j} \bmod p$ and $h_i = g^{ss_i} \bmod p$. The public key is $H = \prod_{i=1}^{n} h_i \cdot \prod_{j=1}^{m} h_j \bmod p$, and the private key is $S = \sum_{i=1}^{n} sb_i + \sum_{j=1}^{m} ss_j$. Note that the sb_j, $1 \leq j \leq m$ and the ss_i, $1 \leq i \leq n$ are kept secret by each individual buyer and seller respectively, and they never share their knowledge due to the the conflict of interests among them. Therefore, S cannot be retrieved unless with the cooperation of all buyers and sellers, which is impossible.

After computation of the public key H, the buyers/sellers generate their bids/asks in the specific form, as follows [7]. For buyers, the bid b_j, $1 \leq j \leq m$, which represents price $p_t (1 \leq t \leq K)$, is a vector of K ElGamal encryption texts($E(z)$ or $E(1)$)[5] (we omit the module of p in the formula for clarity of presentation):

$$b_j = (\underbrace{b_{(j,1)}, b_{(j,2)}, , \ldots, b_{(j,t)}}_{t}, \underbrace{b_{(j,t+1)}, \ldots, b_{(j,K)}}_{K-t})$$

[4] A random tie-breaking is applied here.

[5] Without loss of generality, we assume that the common public element $z = 2$ in context.

where

$$b_{(j,\omega)} = \begin{cases} E(2) = (g^{\delta_{(j,\varpi)}}, H^{\delta_{(j,\varpi)}} \cdot 2),\ if\ 1 \leq \varpi \leq t \\ E(1) = (g^{\delta_{(j,\varpi)}}, H^{\delta_{(j,\varpi)}} \cdot 1),\ if\ t < \varpi \leq K \end{cases}$$

Here, $\delta_{(j,\varpi)}(1 \leq \varpi \leq K)$ are K numbers arbitrarily selected by buyer B_j. Note that the $\delta_{(j,\varpi)}$ should be selected arbitrarily, so the K elements in bid vector b_j are indistinguishable from each other due to the probabilistic property of ElGamal system. Similar to the definition for buyers, the ask a_i, $1 \leq i \leq n$, of the sellers, which represents price $p_t(1 \leq t \leq K)$, is also a vector of K ElGamal encryption texts:

$$a_i = (\underbrace{a_{(i,1)}, a_{(i,2)}, , ..., a_{(i,t-1)}}_{t-1}, \underbrace{a_{(i,t)}, ..., a_{(i,K)}}_{K-t+1})$$

where

$$a_{(i,\omega)} = \begin{cases} E(1) = (g^{\gamma_{(i,\varpi)}}, H^{\gamma_{(i,\varpi)}} \cdot 1),\ if\ 1 \leq \varpi \leq t-1 \\ E(2) = (g^{\gamma_{(i,\varpi)}}, H^{\gamma_{(i,\varpi)}} \cdot 2),\ if\ t < \varpi \leq K \end{cases}$$

Here, $\gamma_{(i,\varpi)}$, $1 \leq \varpi \leq K$ are K numbers, arbitrarily selected by seller S_i (note the reverse order of the element in a_i).

Next, buyers/sellers publish their bids sb_j and asks sa_i. To provide public verifiability, each buyer/seller must prove the correctness of his bid/ask, *i.e.* that it suits the form described, in zero-knowledge manner (refer to **Appendix 1**).

Similar to that of [7], the auctioneer then publicly calculates the component-wise product of all bid sb_j vectors and asks vector sa_i, respectively. We only present the calculation for buyers as follows, as that for sellers is similar.

The product of all b_j is,

$$\prod_{j=1}^{m} b_j = (\prod_{j=1}^{m} b_{(j,1)}, ..., \prod_{j=1}^{m} b_{(j,K)}) = (B^{(1)}, B^{(2)}, ..., B^{(K)})$$

where

$$\begin{aligned} B^{(f)} &= \{\prod_{j=1}^{m} (g^{\delta_{(j,f)}}),\ 2^{n_{(b,f)}} \cdot \prod_{j=1}^{m} H^{\delta_{(j,f)}}\} \\ &= \{g^{\sum_{j=1}^{m} \delta_{(j,f)}},\ 2^{n_{(b,f)}} \cdot H^{\sum_{j=1}^{m} \delta_{(j,f)}}\} \\ &= \{g^{\Omega}, 2^{n_{(b,f)}} \cdot H^{\Omega}\} \end{aligned}$$

where $\Omega = \sum_{j=1}^{m} \delta_{(j,f)}$. To calculate the $n_{(b,f)}$, $1 \leq f \leq K$ (*i.e.*, the number of buyers whose bid equal to or higher than p_f), all buyers and sellers perform the distributed ElGamal decryption on $B^{(f)}$ as presented in section 3.2. That is, each buyer B_j computes and publishes $x_{B_j} = (g^{\Omega})^{sb_j}$, with $1 \leq j \leq m$ and each seller S_i computes and publishes $x_{S_i} = (g^{\Omega})^{ss_i}$, with $1 \leq i \leq n$, respectively.[6] From all x_{B_j} and x_{S_i}, the auctioneer computes (recall that $H = \prod_{i=1}^{n} h_i \cdot \prod_{j=1}^{m} h_j = g^{(\sum_i ss_i + \sum_i sb_j)}$)

$$\frac{2^{n_{(b,f)}} \cdot H^{\Omega}}{\prod_{i,j} x_{B_j} \cdot x_{S_i}} = \frac{2^{n_{(b,f)}} \cdot H^{\Omega}}{g^{\Omega(\sum_i ss_i + \sum_i sb_j)}} = \frac{2^{n_{(b,f)}} \cdot g^{\Omega(\sum_i ss_i + \sum_i sb_j)}}{g^{\Omega(\sum_i ss_i + \sum_i sb_j)}} = 2^{n_{(b,f)}}$$

Clearly, with a simple computation of base-2 logarithm of the above value,[7] the auctioneer can get $n_{(b,f)}$. Then, the total demand of the double auction market can be deduced as follows,

$$\begin{array}{l} N_{(b,p_K)} = n_{(b,K)} \\ N_{(b,p_f)} = n_{(b,f)} - n_{(b,f+1)} \end{array}, \text{ for } 1 \leq f < K$$

Here, $N_{(b,p_f)}$, $1 \leq f \leq K$, is the number of buyers whose bid price equal to p_f. In a similar way, the auctioneer can calculate the $n_{(a,f)}$, which is the number of sellers whose asks price equal to or lower than p_f.

By computing:

$$\begin{array}{l} N_{(a,p_1)} = n_{(a,1)} \\ N_{(a,p_f)} = n_{(a,f)} - n_{(a,f-1)} \end{array}, \text{ for } 1 < f \leq K,$$

the auctioneer gets the total supply of the market $N_{(a,p_f)}$, $1 \leq f \leq K$. From $N_{(b,p_f)}$ and $N_{(a,p_f)}$, the auctioneer can always list two ordered sequences of all bids $b_{(j)}$ and asks $a_{(i)}$ like what PMD protocol does, even though he does not know what each bid/ask price actually is.

$$\begin{array}{l}
\overbrace{a_{(1)} = ... = a_{(N_{(a,p_1)})}}^{N_{(a,p_1)}} < \overbrace{... = a_{(N_{(a,p_1)}+N_{(a,p_2)})}}^{N_{(a,p_2)}} < ... < \overbrace{... = a_{(N_{(a,p_1)}+...+N_{(a,p_K)})}}^{N_{(a,p_K)}} \\
\underbrace{b_{(1)} = ... = b_{(N_{(b,p_K)})}}_{N_{(b,p_K)}} > \underbrace{... = b_{(N_{(b,p_K)}+N_{(b,p_{K-1})})}}_{N_{(b,p_{K-1})}} > ... > \underbrace{... = b_{(N_{(b,p_K)}+...+N_{(b,p_1)})}}_{N_{(b,p_1)}}
\end{array}$$

Note that $\sum_{f=1}^{K} N_{(a,p_f)} = n$ and $\sum_{f=1}^{K} N_{(b,p_f)} = m$. Here, $a_{(n)}$ is highest *ask* of sellers, and $b_{(m)}$ the lowest *bid* of buyers.

Thus, the auctioneer gets the total demand and supply information of the market while knowing nothing about the bid/ask of individual buyer/seller. The

[6] Note that each buyer/seller should prove the correctness of the x_{B_j} (x_{S_i}) in a zero-knowledge manner, *i.e.*, Chaum's proof of equality of discrete logarithms [13] of x_{B_j} (x_{S_i}) and $h_j(h_i)$.

[7] To make the computation feasible, we require that $2^{n_{(b,f)}}$ be less than the module p of ElGamal algorithm. For example, the maximal number of sellers/buyers should be less than 1024 if we employ a 1024-bit ElGamal system.

following calculation of the trading price is the same as that of PMD protocol. That is, the auctioneer first chooses a t such that $a_{(t)} \leq b_{(t)}$ and $a_{(t+1)} > b_{(t+1)}$, then defines the candidate of a trading price $p_0 = \frac{1}{2}(b_{(t+1)} + a_{(t+1)})$. The protocol is described as follows.

- If $p_0 \in [a_{(t)}, b_{(t)}]$, the buyers/sellers from (1) to (t) trade at price p_0. In other words, those buyers whose bids equal to or higher than b_t trade with those sellers whose asks equal to or lower than a_t.
- If $p_0 \notin [a_{(t)}, b_{(t)}]$, only buyers/seller from (1) to $(t-1)$ trade, which means that those buyers whose bids equal to or higher than $b_{(t-1)}$ trade with those sellers whose asks equal to or lower than $a_{(t-1)}$. Each buyer pays $b_{(t)}$, and each seller gets $a_{(t)}$.

To determine the winning buyers (*i.e.* whose bids equal to or higher than certain price p_f) without revealing their *bids*, all buyers and sellers perform distributed ElGamal decryption on the f-th element $b_{(j,f)}$ of the each bid vector b_j, and find the winning buyers B_j with $D(b_{(j,f)}) = 2$.[8] In a similar way, we can find the winning sellers (*i.e.* whose asks equal to or lower than certain price $p_{f'}$) without revealing their asks. That is, all buyers and sellers perform distributed ElGamal decryption on the f'-th element $a_{(i,f')}$ of the each ask vector a_i, and find the winning sellers S_i with $D(a_{(i,f')}) = 2$.

3.4 A Secure Double Auction Protocol Based on Yokoo's Protocol

As an extension to the PMD protocol, a robust double auction protocol (called TPD) is presented by Yokoo *et al.*[10] against false-name bids. We first review the TPD protocol in short.

In the TPD protocol, the auctioneer first determines a so-called *threshold price* r.[9] Buyers/sellers then declare their bids/asks. Let these bids and asks be ordered as follows.

$$\text{Buyers' bids: } b_{(1)} \geq b_{(2)} \geq \ldots \geq b_{(i)} \geq r \geq b_{(i+1)} \geq \ldots \geq b_{(m)}$$

$$\text{Sellers' asks: } s_{(1)} \leq s_{(2)} \leq \ldots \leq s_{(j)} \leq r \leq s_{(j+1)} \leq \ldots \leq s_{(n)}$$

The TPD protocol is defined as follow.

- When $i = j$: the buyers/seller from (1) to (i) trade at the price r.
- When $i > j$: the buyers/sellers from (1) to (j) trade. Each buyer pays $b_{(j+1)}$ and each seller gets r. The auctioneer gets the amount of $j \cdot (b_{(j+1)} - r)$.
- When $i < j$: the buyers/sellers from (1) to (i) trade. Each buyer pays r and each seller get $s_{(i+1)}$. The auctioneer gets the amount of $i \cdot (r - s_{(i+1)})$.

[8] Here, $D(m)$ means the distributed ElGamal decryption on m.

[9] Yokoo *et at.* have discussed the choice of r in [10], and it is shown that the best result (*i.e.* that achieving the maximal social welfare) is obtained when r is the median of all possible bid/ask prices. In our paper, we assume that the auctioneer may choose any price in $[p_1, p_K]$ as the threshold price.

The TPD protocol is the first non-trivial double auction protocol which is proven to be a *dominant-strategy incentive compatible* protocol even if participants might submit false-name bids. Based on the TPD, we propose a secure double auction protocol with full privacy protection of participants. As described in section 3.2, the buyers/sellers generate the open public key H, and publish the sealed bids/asks vector of ElGamal ciphertexts. Also, the auctioneer calculates the component-wise product of all bids b_j vectors and asks vectors a_i, respectively, as that in section 3.2.

The product of all b_j is

$$\prod_{j=1}^{m} b_j = (\prod_{j=1}^{m} b_{(j,1)}, \ldots, \prod_{j=1}^{m} b_{(j,K)}) = (B^{(1)}, B^{(2)}, \ldots, B^{(K)}),$$

where

$$B^{(f)} = \{g^{\Omega},\ 2^{n_{(b,f)}} \cdot H^{\Omega}\}$$

The product of all a_i is

$$\prod_{j=1}^{n} a_i = (\prod_{i=1}^{n} a_{(i,1)}, \ldots, \prod_{i=1}^{n} a_{(i,K)}) = (A^{(1)}, A^{(2)}, \ldots, A^{(K)}),$$

where

$$A^{(f)} = \{g^{\Omega},\ 2^{n_{(a,f)}} \cdot H^{\Omega}\}$$

All buyers/seller now perform the distributed ElGamal decryption on $B^{(\lambda)}$ and $A^{(\lambda)}$ to calculate the $n_{(b,\lambda)}$ and $n_{(a,\lambda)}$. Note that p_λ is the threshold price r chosen by auctioneer here. Clearly, $n_{(b,\lambda)}$ is the number of buyers whose bids equal to or higher than r, and $n_{(a,\lambda)}$ is the number of sellers whose asks equal to or lower than r. There are three possible situations.

- When $n_{(b,\lambda)} = n_{(a,\lambda)}$, buyers whose bids equal to or higher than r and sellers whose asks equal to or lower than r trade at price r. The method to determine the winning buyers/sellers is as same as that in section 3.3.
- When $n_{(b,\lambda)} > n_{(a,\lambda)}$, there are $n_{(a,\lambda)}$ trades occur. Those buyers whose bids equal to or higher than $b_{(n_{(a,\lambda)})}$ are eligible to trade and pay $b_{(n_{(a,\lambda)}+1)}$. Those sellers whose asks lower than r are eligible to trade and get r. The method to determine the winning sellers is the same as described before. To determine the winning buyers is similar to solving the problem of the $(M+1)$-st price auction [6,7], with $M = n(a,\lambda)$. Here, we apply the method proposed by Abe *et al.*[7]. By applying the mix and match technique [14] to $B^{(f)}$, $1 \leq f \leq K$, the auctioneer can examine whether $n_{(b,f)} \leq n_{(a,\lambda)}$, without knowing the $n_{(b,f)}$ itself (refer to **Appendix 2** for details). To find out the threshold winning price $b_{(n_{(a,\lambda)})}$ for buyers, *i.e.*, price p_w such that $n_{(b,w)} \leq n_{(a,\lambda)}$ and $n_{(b,w+1)} \geq n_{(a,\lambda)} + 1$, the auctioneer performs a binary search (between p_K and r) on $B^{(f)}$, $1 \leq f \leq K$, using the examination by mix and match. On finding the p_w, we can apply the same method described before to determine the $n_{(a,\lambda)}$ winning buyers.

- When $n_{(b,\lambda)} < n_{(a,\lambda)}$, there are $n_{(b,\lambda)}$ trades occuring. Those buyers whose bids are higher than r are eligible to trade and pay r. Those sellers whose asks are lower than $a_{(n_{(b,\lambda)})}$ are eligible to trade and get $a_{(n_{(b,\lambda)}+1)}$. The method to determine the winning buyers/seller is the same as what described in the second situation above.

4 Analysis of the Protocols

In this section, we perform an analysis of the protocols with respect to security and efficiency.

4.1 Security

Collusion-Proof of the Protocols. The proposed secure double auction protocols aim to protect the privacy of buyers/sellers. The following theorem states that with these protocols, the bids and asks are always kept secret.

Theorem *In two proposed protocols, the sensitive information (mentioned in section 2.2) regarding the bid/ask is always kept secret even in the presence of any collusion of other buyers/sellers and auctioneer.*

Firstly, recall that in proposed protocols the sealed bid/ask is a vector of K ElGamal encrypted texts, say a bid b which represents price p_j $(1 \leq j \leq K)$

$$b = (b^{(1)}, b^{(2)}, ..., b^{(K)}) = (\underbrace{E(z), ..., E(z)}_{j}, \underbrace{E(1), ..., E(1)}_{K-j})$$

Determining what price b represents is equivalent to determining what j is. Due to the indistinguishability of ElGamal encryption algorithm, no one can determine the value of j without decrypting each elements of the vector. In our protocol, we do not assume the existence of any trusted authorities. Instead, we distribute the trust over all buyers and sellers. That is, each buyer/seller generates a secret number ss^i/sb^j independently, and then a public key H is generated as $H = g^{\sum_i ss^i + \sum_j sb^j}$ for encryption on each element $b^{(i)}$ in bid b. As a result, the only way to decrypt any element of bid b is that all buyers/sellers participate in the computing with their secret numbers, respectively. Therefore, any group of colluding buyers/sellers cannot open any specific sealed bid b_j/ask a_i, without the cooperation of all other buyers/sellers.

Furthermore, to determine the winning buyers/sellers (*i.e.* whose bids/asks are higher/lower than certain price p_f) in our protocols, we only require that all buyers/sellers perform the distributed decryption on the f-th element of each sealed bid/ask. In this way, the only information revealed is that whether the bid/ask is higher/lower than a certain threshold price, while the privacy of the bid/ask is still protected.

Table 1. Communication complexity of PMD based secure DA protocol

Secure DA protocol based on McAfee's one	*Pattern*	*round*	*volume*
Submitting bids/asks (Per one buyer/seller)	Buyer/seller→Auctioneer	1	$O(K)$
Determining transaction price (Per one buyer/seller)	Auctioneer→Buyer/seller	1	$O(K)$
Determining winning buyer/seller	Auctioneer→Buyer/seller	1	$O(m+n)$

Table 2. Communication complexity of TPD based secure DA protocol

Secure DA protocol based on Yokoo's one	*pattern*	*round*	*volume*
Submitting bids/asks (Per buyer/seller)	Buyer/seller→Auctioneer	1	$O(K)$
Determining transaction price (Per buyer/seller)	Auctioneer→Buyer/seller	$\lceil \log(\lambda) \rceil + 1$	$O(M+1)$
Determining winning buyer/seller	Auctioneer→Buyer/seller	1	$O(m+n)$

Privacy Protection. Note that, in the first secure double auction protocol that is based on the PMD protocol, the statistic information of bids/asks (*i.e.* the total supply and demand of the market) has to be open, though corresponding information between bids/asks and the identities of buyers/sellers is protected. This is what we call *weak privacy* protection. In the second proposed protocol based on the TDP protocol, *strong privacy* protection is achieved. That is, only the transaction price is revealed at the end of the protocol, while the privacy of the bid/ask as well as the statistic information of bids/asks are protected.

Public Verifiability and Weak Robustness. It seems difficult to assure robustness of our protocols, since we distribute the secure computation on all buyers/sellers. Our protocols, however, are publicly verifiable, which means that every step of the protocols can be publicly verified by any participants in the auction. Such verifiability can be used to provide the so-called *weak robustness*, so that malicious bidders will be detected immediately without additional communication and information revelation and can be excluded from the set of participants.

4.2 Efficiency

We discuss the communication efficiency and the computation efficiency of the proposed protocols in this section.

Tables 1 and 2 show the communication complexity of two proposed protocols when there are m buyers, n sellers and K potential prices (note that, in tables

Table 3. Computation complexity of PMD based secure DA protocol

Secure DA protocol based on McAfee's one	*Computational complexity*
Public key generation	One modulo exponential computation and one proof for each buyer/seller
Bid/Ask generation	K encryptions and proofs for each buyer/seller
Determining transaction price	$2(m+n)K$ ciphertext multiplications for auctioneer; K modulo exponential computations and proofs for each buyer/seller
Determining winning buyer/seller	$(m+n)$ ciphertext multiplications for auctioneer; $(m+n)$ modulo exponential computation and proofs for each buyer/seller

Table 4. Computation complexity of TPD based secure DA protocol

Secure DA protocol based on Yokoo's one	*Computational complexity*
Public key generation	One modulo exponential computation and one proof for each buyer/seller
Bid/Ask generation	K encryptions and proofs for each buyer/seller
Determining transaction price	$(m+n)(K+\lceil\log(\lambda)\rceil+1)$ ciphertext multiplications and $\lceil\log(\lambda)\rceil$times (M+1)mix and match for auctioneer; $\lceil\log(\lambda)\rceil$(M+1)+1 modulo exponential computations and proofs for each buyer/seller
Determining winning buyer/seller	$(m+n)$ ciphertext multiplications for auctioneer; $(m+n)$ modulo exponential computations and proofs for each buyer/seller

2 and 4 $M = Min(n_{(a,\lambda)}, n_{(b,\lambda)})$ where p_λ is the threshold price). It is shown that our protocols have low round communication complexity: one round for any phase in first protocol; one round for bid/ask submitting, determining winning buyers/sellers and $\lceil\log(\lambda)\rceil+1$ rounds for determine the transaction price in the second protocol.

Tables 3 and 4 show the computational complexity of our protocols. It is clear that the computational complexity of both proposed protocols is approximately determined by K and $m+n$, so our schemes may have heavy cost implication in case of a large price range or a large number of buyers/sellers. Actually, we require that the number of buyers/sellers should be less than the module p of the ElGamal system. Therefore, a reasonable price interval should be applied for an efficient implementation of our protocols.

5 Conclusion

In this paper, we first consider the security issues in double auction system, and propose two secure double auction protocols based on PMD protocol and TPD protocol, respectively. The proposed protocols achieve full privacy protection of buyers/sellers and the public verifiability. In our protocols, we do not make any special trusted assumption, and full privacy protection here means that all sensitive information regarding the bids/asks, as we mentioned in section 2.2, is always kept secret no matter any collusion of other buyers/sellers and auctioneers[10].

Furthermore, our protocols have low round communication complexity and a reasonable computation complexity. In general, the complexity of our protocols is proportional to the number of the possible offer prices and the total number of the participants. So our protocols may not be applicable when the price step is too small. Instead, the auctioneer can first divide a large-scale market into several small ones and then follow the protocols respectively and finally gather the total result one by one.

Acknowledgement

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK 4346/02E).

References

[1] Friedman, D. and Rust, J. (eds).:The Double Auction Market: Institutions, Theories and Evidence. Addison-Wesley 1992.

[2] Matthew K. Franklin and Michael K. Reiter.: The design and implementation of a secure auction server. *IEEE Trans. on Software Engineering*, 22(5) pp 302-312, 1996

[3] Lipmaa H., Asokan N., and Niemi V.: Secure Vickrey auctions without threshold trust. In *Proceedings of the 6th Annual Conference on Financial Cryptography*, 2002.

[4] Kikuch H., Harkavy M. and Tygar J. D.: Multi-round anonymous auction protocols. *Proceedings of the First IEEE Workshop on Dependable and Real-time E-Commerce systems*, pp 62-69, 1998.

[5] Cachin C.: Efficient private bidding and auctions with an oblivious third party. In *proceedings of the 6th ACM Conference on Computer and Communications Security*, pp 120-127, 1999.

[6] Kikuchi H.: (M+1)st-price auction protocol. In *proceedings of Financial Cryptography (FC2001)*, 2001.

[10] Note that, the first secure double auction protocol based on the PMD protocol only achieves the *weak privacy* protection, since the total market supply and demand information has to be revealed to calculate the transaction price. In such case, any $m-1$ buyers or $n-1$ sellers can always collude to retrieve the another one's bid.

[7] Abe M. and Suzuki K.: (M+1)-st price auction using homomorphic encryption. In *Proceedings of the 5th International Conference on Public Key Cryptography (PKC-02)*, 2002.
[8] McAfee Preston R.: A dominant strategy double auction. *Journal of Economic Theory*, (56), pp 434-450, 1992.
[9] Wurman P R, Walsh E W, Wellman P M.: Flexible Double Auctions for Electronic Commerce: Theory and Implementation. Decision Support Systems, 1998, 24: pp17-27.
[10] Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara.: Robust Double Auction Protocol against False-name Bids. In *Proceedings of 21st IEEE International Conference on Distributed Computing Systems (ICDCS-2001)*, pp 137-145, 2001.
[11] Vickrey W.: Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance*, pp 8-37, 1961.
[12] Schnorr, C. P.: Efficient signature generation by smart cards. *Journal of Cryptology*, 4, pp 161-174, 1991.
[13] Chaum, D., Pedersen, T. P.: Wallet databases with observers. In: *Advances in Cryptology – Proceedings of the 12th Annual International Cryptology Conference*, LNCS 740, Springer Verlag, 1992.
[14] Jakobsson M. and A. Juels.: Mix and Match: Secure Function Evaluation via Ciphertexts. *In Advances in Cryptology - ASIACRYPT 2000*, LNCS 1976, SpringerVerlag, pp162-177, 2000.
[15] Abe M.: Mix-Networks on Permutation Networks. Proceedings of ASIACRYPT'99 PP 317-324, 1999
[16] ElGamal, T., A Public-key cryptosystem and a signature scheme based on discret logarithms, *In Advances in Cryptology - CRYPTO'84 Proceedings*, SpringerVerlag, pp10-18, 1985.

Proof of the Correctness of Bid/Ask Vector

To provide public verifiability, each buyer/seller must prove the correctness of his bid/ask vector, i.e. it suits the form described, in zero-knowledge manner. That is, the buyer/seller should prove that the submitted vector comprises K factors, each of which is a ElGamal ciphertext of either 1 or common parameter z, while the verifier can not confirm whether the ciphertext is of 1 or z. We give the general zero-knowledge proof of such problem as follows.

Proof . Let a factor of bid/ask vector $E(m)$ be a suit of data $E(m) = (x, y) = (g^\alpha \bmod p, H^\alpha m \bmod p)$, $m \in \{G_0, G_1, ..., G_W\}$[11], where g and H are public parameters, α and m are kept secret. To prove the correctness of $E(m)$, the buyer/seller generate a proof data f (assume that $m = G_k$):

$$f = \{\ (s, C, C_0, C_1, \ldots, C_{k-1}, C_k, C_{k+1}, \ldots, C_W), \\ (u_0, u_1, \ldots u_{k-1}, u_k, u_{k+1}, \ldots, u_W), (\beta)\ \},$$

where,

(1) $s, C_0, C_1, \ldots, C_{k-1}, C_{k+1}, \ldots, C_K$ are randomly selected integers;

[11] Actually, in this paper, $m \in \{1, 2\}$. Here, we give a general proof of such zero-knowledge proof problem.

(2) $u_i = \frac{H^a \cdot m}{g^a \cdot G_i} = A^a \cdot \frac{G_k}{G_i}, \quad i = (0, ..., W)$, Here$A = \frac{H}{g}$. Note that $u_k = A^a$.
(3) $C = H(A^s \cdot u_0^{C_0} \cdot u_1^{C_1} ... \cdot u_{k-1}^{C_{k-1}} \cdot u_{k+1}^{C_{k+1}} \cdot ... \cdot u_K^{C_K})$
(4) $C_k = C \oplus C_1 \oplus ... \oplus C_{k-1} \oplus C_{k+1} \oplus ... \oplus C_K$
(5) $\beta = s - C_k \cdot a$

Verification. Anyone can verify the correctness of $E(m)$, by checking whether the equation: $C_1 \oplus ... \oplus C_K = H(A^{\beta} \cdot u_0^{C_1} \cdot ... \cdot u_K^{C_K})$ hold.

Mix and Match Technology on An ElGamal Ciphertext

By using the mix and match technology on $B^{(f)} = E(2^{n(f)}),\ 1 \leq f \leq K$, we can examine whether $n(f) \leq \rho$[14] or not, without knowing the $n(f)$ itself. That is to examine whether $D(B^{(f)}) \in \{1, 2, 2^2, ..., 2^{\rho}\}$ or not. Note that, $E()$ and $D()$ means the distributed ElGamal encryption and decryption, respectively.

First, we construct ρ ciphertexts $B_i = B^{(f)}/E(2^i),\ 0 \leq i \leq \rho$, and take the power $B_i^{r_i}$of each ciphertext using a secret random factor r_i, thus mix them. Next, we make the distributed ElGamal decryption on the mixed ρciphertexts $B_i^{r_i}$. If there exists one plaintext 1, it is convinced that $n(f) \leq \rho$; if there exists no plaintext 1, it is convinced that $n(f) > \rho$.

Sealed-Bid Auctions with Efficient Bids

Toru Nakanishi, Daisuke Yamamoto, and Yuji Sugiyama

Department of Communication Network Engineering,
Faculty of Engineering, Okayama University
3-1-1 Tsushima-naka, Okayama 700-8530, Japan
{nakanisi,daisuke,sugiyama}@cne.okayama-u.ac.jp

Abstract. Efficient general secure multiparty computation (MPC) protocols were previously proposed, and the combination with the efficient auction circuits achieves the efficient sealed-bid auctions with the full privacy and correctness. However, the combination requires that each bidder submits ciphertexts of bits representing his bid, and their zero-knowledge proofs. This cost amounts to about 80 multi-exponentiations in usual case that the bid size is 20 bits (i.e. about 1,000,000 bid prices). This paper proposes the sealed-bid auction protocols based on the efficient MPC protocols, where a bidder can submit only a single ciphertext. The bidder's cost is a few multi-exponentiations, and thus the proposed protocols are suitable for mobile bidders.

Keywords. Sealed-bid auction, multiparty computation (MPC), mix-and-match, additive homomorphic cryptosystems

1 Introduction

1.1 Backgrounds

The Internet has been providing a commercial environment, where secure auctions are in demand. A usual type of them is a sealed-bid auction. In the sealed-bid auction, bidders submit their own bid prices to the auctioneer secretly, until the deadline. After the deadline, the auctioneer announces the winner and the clearing price. In the highest-price auction, the bidder who submitted the highest bid is the winner, and the clearing price is the highest bid. On the other hand, in the second-price auction known Vickrey auction, the bidder with the highest one wins but the clearing price is the second highest bid. Furthermore, the Vickrey auction is extended into the $(M+1)$-st auction [16]. This type assumes that multiple units of an item are auctioned. Then, the winners are M highest bidders, and the clearing price is the $(M+1)$-st highest bid.

In the sealed-bid auction, two important properties are required. One is the bid privacy. The minimum is the privacy until the deadline. Additionally, it is desired that the privacy continues even after the deadline, as pointed out in [13], since the information on the bids (i.e., prices) indicates bidders' preferences or powers. Furthermore, the statistics of bids in an auction help the auctioneer to learn the strategy of the bidders, which is useful for managing other auctions.

This type of strong privacy is called full privacy. The other important property is the correctness of the winners and the clearing price. This is required to prevent the auctioneer's cheating.

1.2 Related Works

A straightforward solution is the unquestioning trust of the auctioneer. Bidders submit ciphertexts of their bids, and the auctioneer decrypts them to decide the winners. To reduce the trust, a cryptographic method known as the general secure multiparty computation (MPC) [12] can be applied. The MPC assumes multiple servers. Generally, the servers are trusted in threshold fashion, namely it is trusted that any quorum of the servers is honest. On the trust, the MPC targets that the servers cooperatively and correctly evaluate a circuit for any function while the given inputs or any other information are kept secret. Then, any type of sealed-bid auction with the full privacy and correctness can be realized by the MPC. However, when the MPC based on verifiable secret sharing schemes (VSS) as [12] is applied to the auctions, each bidder, who gives the secret input of the function, is required heavy computations concerning the VSS.

As another stream, some protocol designs peculiar to auctions have been proposed [13, 4, 16, 2, 20] with the full privacy and correctness. However, these solutions have the problem of the computation complexity or the privacy. The schemes [13, 16] adopt the VSS for the robustness, and thus heavily load on the bidders, as the MPC on the VSS. Let k be the length of a bid. Then, the schemes [2, 20] require $O(2^k)$ computation complexity for bidders. The scheme [4] involving with two servers requires only $O(k)$, but a server gains the information: a partial order of the bids.

On the other hand, recently, two efficient MPC protocols have been proposed [14, 7], which avoid the use of the VSS. The common essential idea is the use of threshold cryptosystems, where a private decryption key is shared by the servers. Then, for a circuit computing a target function, given the encrypted inputs, the servers evaluate each gate to make the encrypted output, and thus the evaluations of all the gates in turn lead to the evaluation of the circuit. Jakobsson and Juels [14] proposed an MPC protocol, called mix-and-match, to target the evaluation of a logical circuit, and it is effectively applied to sealed-bid auctions. A boolean circuit to realize a sealed-bid auction is very simple, as explored in [17], and the complexity is $O(k)$. Therefore, this MPC protocol carries out the sealed-bid auction with the complexity $O(k)$. In addition, the inputs of the circuit are the ciphertexts of the bits representing his bid. Thus, a bidder only submits ciphertexts of the bits representing his bid without conducting the VSS. Though this MPC protocol adopts the ElGamal cryptosystem, Cramer et al. [7] proposed another MPC protocol based on the additive homomorphic cryptosystem, such as the Paillier cryptosystem [21]. This protocol targets the evaluation of an arithmetic circuit over a ring. Since such a circuit can simulate any boolean circuit, this protocol can be also applied to the sealed-bid auctions with the complexity $O(k)$, as well as the mix-and-match. As a final note, auctions based on efficient MPC protocols without the VSS, where only two servers participate, have also been proposed [19, 15], but the extensions to protocols involving with more than two servers are unknown.

1.3 Our Contributions

In the auctions based on the efficient MPC protocols [14, 7] in the general threshold trust, a bidder submits ciphertexts of the bids representing his bid. In addition, to exclude incorrect bids for obstructing the auction, the bidder proves the correctness of ciphertexts in zero-knowledge fashion. The ElGamal encryption and its proof requires 4 multi-exponentiations. Hence, when the realistic parameter $k = 20$ is used, i.e., the bid can represent about 1,000,000 ($\approx 2^{20}$) prices, then the bidder is required to compute 80 multi-exponentiations. This cost is vast for mobile bidders with low computation powers. The case of the additive homomorphic encryption such as the Paillier encryption is similar. This paper proposes auction protocols based on the previous MPC protocols, where a bidder only submits a single ciphertext of a message representing his bid. Thus, the bidder's cost is $O(1)$ and a few multi-exponentiations, which is more suitable to the mobile bidder. On the other hand, the servers cooperatively convert the received ciphertext into the ciphertexts of the bits representing the bid. This computation preserves the correctness and privacy with the robustness. In addition, the cost is $O(k)$, and efficient. After the conversion, the evaluation of any type of sealed-bid auction circuit is achieved by the previous MPC protocols. This paper extends both type of MPC protocols based on the mix-and-match and on the additive homomorphic cryptosystem. This is because both types can be refined by the improvements of underlying cryptographic tools and thus it is hard to compare both types and to pick out one type.

Hereafter, this paper is organized as follows: The next section reviews the model and requirements of sealed-bid auctions. Then, in Section 3, we first extends the MPC protocol on additive homomorphic cryptosystem, since the cryptosystem is also utilized in the next section. In Section 4, we will explore the mix-and-match type.

2 Model and Requirements

We adopt the model where the multiple servers cooperatively play the role of the auctioneer. Assume that, among all L servers, any quorum indicated by a threshold value is honest. Then, the requirements of sealed-bid auctions are as follows [14]:

Full privacy: The revealed information of an auction is only the the identities of winners and the clearing price, unless any quorum of the servers collude. For example, the information in the highest-price auction is the identity of the highest bidder and his bid price.

Public verifiability: The correctness of the auction can be verified by any party (even any external party).

Robustness: An adversary corrupting some bidders and servers who do not make the quorum cannot disrupt the auction protocol and undermine the full privacy.

Auction adaptability: Any type of a sealed-bid auction, i.e., a highest-price auction, a second-price auction, and an $(M+1)$st-price auction, can be incorporated.

3 Sealed-Bid Auction with Efficient Bids Using Additive Homomorphic Cryptosystem

This section extends the sealed-bid auction based on the MPC protocol [7] for additive homomorphic cryptosystem. In the extension, a bidder submits only a single ciphertext.

3.1 Building Blocks

The proposed protocol adopts any additive homomorphic public-key cryptosystem with some properties. By $E(m)$, we denote a ciphertext on plaintext m with a randomness. Let N be the modulus to define the plaintext space $\mathbb{Z}_N$, where $|N|$ is denoted by ℓ_N. For an integer a and a positive integer d, let $[a, a+d]$ be the integer interval of all *int* s.t. $a \leq int \leq a+d$. We often use the notation $[a, a+d)$ which is the integer interval of all *int* such that $a \leq int < a+d$. In addition, let $\in_R$ denote the uniform random selection. For any bit-string x, $x[i]$ denotes the $(i+1)$-st bit from LSB of x, and $x = (x_{k-1}, \ldots, x_0)_2$ denotes k-bit integer, where $x_i = x[i]$ with all $0 \leq i \leq k-1$. Then, we need the following properties:

Additive homomorphic: There exists an efficiently computed operation $\otimes$ s.t. $E(m_1) \otimes E(m_2) = E(m_1 + m_2 \bmod N)$. Additionally, there exists an efficiently computed operation $E(m)^{-1}$ s.t. $E(m)^{-1} = E(-m \bmod N)$.

Constant multiplication: There exists an efficiently computed operation that, on inputs the public key, a ciphertext $E(m_1)$, a plaintext m_2, outputs $E(m_1 m_2 \bmod N)$. Let $E(m_1)^{m_2}$ denote this operation.

Verifiable threshold decryption: The private key can be shared by multiple servers, where any quoram of the servers can cooperatively decrypt a ciphertext, while any set of the servers who does not constitute the quoram has no information on the plaintext. The correctness of the decryption can be verified by any party. Moreover, the robustness holds.

PK of plaintext: There exists a zero-knowledge proof of knowledge (PK), where, on common inputs the public key and $C = E(m)$, a prover can prove the knowledge m s.t. $C = E(m)$.

PK of plaintext in interval: There exists a PK, where, on common inputs the public key and $C = E(m)$, a prover can prove the knowledge m s.t. $C = E(m)$ and $m = [a, a+d]$.

MPC multiplication: There exists an MPC protocol, where, given $E(m_1)$ and $E(m_2)$, the above servers cooperatively compute $E(m_1 m_2 \bmod N)$ without revealing any information on m_1, m_2. The correctness of the multiplication can be verified by any party. Moreover, the robustness holds.

Size and Factorization of N: $|N|$ that is the size of the plaintext space is comparable with the security parameter. For example, $|N| = 1024$. Additionally, N has only large prime factors.

As shown in [7], using the homomorphic property and the MPC multiplication, the MPC protocol for evaluating any boolean circuit (strictly arithmetic circuit) is realized. Thus, by incorporating with an efficient circuit for any type of auction as shown in [17], we obtain the MPC protocol for auctions, called an MPC auction here. In this protocol, as inputs of the circuit, a bidder decides the bid price b with k bits, and has to submit $E(b[0]), \ldots, E(b[k-1])$ to the servers with the verifiability.

Example. An example satisfying the above properties is the Paillier cryptosystem [21]. In the system, the public key is (N, g), where $N = PQ$ is an RSA modulus and g is a random element from $\mathbb{Z}^*_{N^2}$ with the order devided by N, and the private key is the factarization of N. Then, $E(m) = g^m r^N \bmod N^2$ for the randomness r from $\mathbb{Z}_N$. The decryption of $C = E(m)$ is made by $m = L(C^{\lambda(N)} \bmod N^2) \ / L(g^{\lambda(N)} \bmod N^2) \bmod N$, where $L(x) = (x-1)/N$ and $\lambda(N) = \mathrm{lcm}(P-1, Q-1)$. The Paillier cryptosystem is semantically secure, under the decisional composit residuosity assumption (DCRA) [21]. Note that $|N|$ is the security parameter that is currently needed to be 1024, and N has only large prime factors.

For $E(m_1) = g^{m_1} r_1^N \bmod N^2, E(m_2) = g^{m_2} r_2^N \bmod N^2$, $E(m_1) \otimes E(m_2)$ is defined by $E(m_1)E(m_2) \bmod N^2$, which equals $g^{m_1+m_2 \bmod N}\ (r_1 r_2)^N \bmod N^2 = E(m_1 + m_2 \bmod N)$. For $E(m) = g^m r^N \bmod N^2$, $E(m)^{-1}$ is well defined as just the operation, which equals $g^{-m}(r^{-1})^N \bmod N^2 = E(-m \bmod N)$. Similarly, for $E(m_1) = g^{m_1} r_1^N \bmod N^2, m_2$, the operation $E(m_1)^{m_2}$ is well defined as just the operation, which equals $g^{m_1 m_2 \bmod N}\ (r_1^{m_2})^N \bmod N^2 = E(m_1 m_2 \bmod N)$.

The verifiable threshold decryption is shown in [8]. This protocol has one round, where each server conducts some exponentiations and a PK and broadcasts them. The PK of a plaintext and the PK of a plaintext in an interval are shown in [8, 9]. The MPC multiplication is shown in [7]. The protocol has a few round, calls 2 threshold decryptions, and additionally includes some exponentiations depending on the number of the servers and some PKs.

3.2 Overview

The base of our protocol is the MPC auction. Instead of $E(b[0]), \ldots, E(b[k-1])$ and the PK for the verifiability, the bidder submits only $E(m)$ where m is a message representing b. The servers cooperatively convert it into $E(b[0]), \ldots, E(b[k-1])$ in the MPC fashion with the robustness against the bidder's obstruction. For a parameter ℓ_a, consider $a_0 = (b[\ell_a - 1], \ldots, b[0])_2, \ldots, a_{t-1} = (b[k-1], \ldots, b[(t-1)\ell_a])_2$, called sections. In the conversion, the servers first convert $E(m)$ into $E(\tilde{a}_0), \ldots, E(\tilde{a}_{t-1})$, where $\tilde{a}_0 = a_0, \ldots, \tilde{a}_{t-1} = a_{t-1}$ if $E(m)$ is correctly computed. Next, the servers check $\tilde{a}_i \in [0, 2^{\ell_a})$ that is required for the following step, without revealing $\tilde{a}_i$. Finally, from each $E(\tilde{a}_i)$, the servers compute $E(\tilde{a}_i[0]), \ldots, E(\tilde{a}[\ell_a - 1])$. If $\tilde{a}_i = a_i$, these equal $E(b[(i\ell_a)]), \ldots, E(b[(i+1)\ell_a - 1])$. The final ciphertexts $E(b[0]), \ldots, E(b[k-1])$ are imported into the original MPC auction.

If a malicious bidder submits an incorrectly computed ciphertext, $\tilde{a}_i \neq a_i$ may occur. In this case, the bad form is detected if $\tilde{a}_i \notin [0, 2^{\ell_a})$, and otherwise final ciphertexts $E(\tilde{b}[0]), \ldots, E(\tilde{b}[k-1])$ are imported to the MPC auction. The sliced bits $\tilde{b}[0], \ldots, \tilde{b}[k-1]$ may not be ones intended by the bidder, but the bidder has to be responsible for the bits (He may win the bid price that is not intended). The detail is discussed in Section 3.4.

3.3 Proposed Protocols

Setup. Let ℓ be a security parameter controlling the statistical hinding (e.g., $\ell = 128$). Let ℓ_a and $\ell_{\tilde{M}}$ be integer parameters s.t. $0 < \ell_a < \ell_{\tilde{M}} < \ell_N/2$ and $\ell_a + \ell + L + 1 < \ell_{\tilde{M}}$, where L is the number of servers. Let $t = \lfloor \ell_N / \ell_{\tilde{M}} \rfloor$, and $\ell_a t \geq k$ is also required. Define $\tilde{M} = 2^{\ell_{\tilde{M}}}$, and $A = 2^{\ell_a}$.

For k-bit string $b = (b[k-1], \ldots b[0])_2$, let $f(b)$ be the injection mapping to the following plaintext format: $m = a_0 + a_1\tilde{M} + a_2\tilde{M}^2 + \cdots + a_{t-1}\tilde{M}^{t-1}$, where $a_0 = (b[\ell_a - 1], \ldots, b[0])_2, a_1 = (b[2\ell_a - 1], \ldots, b[\ell_a])_2, \ldots, a_{t-2} = (b[(t-1)\ell_a - 1], \ldots, b[(t-2)\ell_a])_2$ and $a_{t-1} = (0, \ldots, 0, b[k-1], \ldots, b[(t-1)\ell_a])_2$. In this mapping, the bits of b are sliced into sections a_i by ℓ_a bits, where the last section a_{t-1} may have less than ℓ_a bits. Hereafter, for the simplicity, assume that $k = \ell_a t$, i.e., all the sections have ℓ_a bits. Other cases can be derived straightforwardly.

The servers set up publicly known polynomial $g(x)$ s.t. $g(x) = 0 \bmod N$ iff $x \in [0, A)$. This can be obtained by $g(x) = x(x-1)\cdots(x-A+1) \bmod N$, which has the degree A. Furthermore, for all $0 \leq u \leq \ell_a - 1$, the servers set up publicly known polynomial $h_u(x)$ s.t. $h_u(x) = x[u] \bmod N$ for all $0 \leq x < A$. Each $h_u(x)$ is computed by the Lagrange interpolation using A points $(x, x[u])$, and has the degree $A - 1$. This computation requires $x - x'$ is invertible on modulus N for any $x, x' \in [0, A)$, which holds due to the property that N has only very large prime factors, as pointed out in [9]. Note that $h_u(a_i) = b[i\ell_a + u]$, for any $a_i = (b[(i+1)\ell_a - 1], \ldots, b[i\ell_a])_2$ $(0 \leq i \leq t-1)$.

Precomputing. This phase can be precomputed before the auction. However, for each bid, this phase has to be done once.

1. Servers cooperatively generate $E(r_i)$ $(0 \leq i \leq t-1)$, where r_i is a uniformly random plaintext on interval $[0, 2^{\tilde{\ell}})$ that is contributed by all honest servers, where $\ell_a + \ell < \tilde{\ell} \leq \ell_a + \ell + L < \ell_{\tilde{M}}$. Concretely, each S_j conducts the following:
   (a) S_j generates $E(r_{ij})$, where $r_{ij} \in_R [0, 2^{\ell_a + \ell})$.

(b) For the ciphertext $E(r_{ij})$, S_j proves the interval, i.e., $r_{ij} \in [0, 2^{\ell_a+\ell})$.

After the above sub-steps, let Q be the set of indices of the servers proved the interval correctly. Without loss of generality, let $Q = \{1, \ldots, |Q|\}$. Then, each honest server sets $E(r_i) = E(r_{i1}) \otimes \cdots \otimes E(r_{i|Q|})$. Note that $E(r_i) = E(\Sigma_{j \in Q} r_{ij} \bmod N) = E(\Sigma_{j \in Q} r_{ij})$ due to $\Sigma_{j \in Q} r_{ij} < N$, and that $\ell_a + \ell < \tilde{\ell} \le \ell_a + \ell + L < \ell_{\tilde{M}}$ for $\tilde{\ell} = |r_i|$.

Furthermore, each server locally computes $E(r) = E(r_0) \otimes E(r_1)^{\tilde{M}} \otimes \cdots \otimes E(r_{t-1})^{\tilde{M}^{t-1}}$.

2. Similarly, a quorum of the servers cooperatively generates $E(R_i)$ $(0 \le i \le t-1)$, where R_i is a uniformly random plaintext on $\mathbb{Z}_N$ that is contributed by all honest servers. Furthermore, using the MPC protocol for multiplication, a quorum of the servers cooperatively computes $E({R_i}^2 \bmod N), \ldots, E({R_i}^A \bmod N)$.

Bidding. A bidder decides his bid b with k bits, and submits $E(m) = E(f(b))$ to the servers.

Conversion. Here, we show the protocol for a single bid, which should be performed for every bid that needs the conversion. The common input of the servers is $E(m) = E(f(b))$. It is converted to $E(b[0]), \ldots, E(b[k-1])$, as follows:

1. **[Section slicing]** In this step, ciphertexts of sections $E(a_0), \ldots, E(a_{t-1})$ are sliced from $E(m)$.
   (a) Using the precomputed $E(r)$, each server locally computes $E(v) = E(m) \otimes E(r)$.
   (b) By the threshold decryption, a quorum of honest servers cooperatively decrypt $E(v)$ to obtain v.
   (c) From v, each server locally slices randomized sections $v_0 = (v[\ell_{\tilde{M}} - 1], \ldots, v[0])_2, \ldots, v_{t-1} = (v[t\ell_{\tilde{M}} - 1], \ldots, v[(t-1)\ell_{\tilde{M}}])_2$ by $\ell_{\tilde{M}}$ bits. Define $\tilde{a}_0 = v_0 - r_0 \bmod N, \ldots, \tilde{a}_{t-1} = v_{t-1} - r_{t-1} \bmod N$.
   (d) Each server locally computes $E(v_i)$ with a common public randomness such as 0, and computes $E(v_i) \otimes E(r_i)^{-1} = E(v_i - r_i \bmod N) = E(\tilde{a}_i)$ for all i.

   For the completeness, consider the case of correctly computed $E(m) = E(f(b))$. Then, from Step (a),(b), $v = m + r = (a_0 + a_1\tilde{M} + \cdots + a_{t-1}\tilde{M}^{t-1}) + (r_0 + r_1\tilde{M} + \cdots + r_{t-1}\tilde{M}^{t-1}) = (a_0 + r_0) + (a_1 + r_1)\tilde{M} + \cdots + (a_{t-1} + r_{t-1})\tilde{M}^{t-1}$. Additionally, $a_0, \ldots, a_{t-1} \in [0, 2^{\ell_a})$ and $r_0, \ldots, r_{t-1} \in [0, 2^{\tilde{\ell}})$, and thus $(a_0 + r_0), \ldots, (a_{t-1} + r_{t-1}) \in [0, \tilde{M})$ due to $\ell_a < \tilde{\ell}$ and $\tilde{\ell} + 1 \le \ell_a + \ell + L + 1 < \ell_{\tilde{M}}$. Therefore, $v_0 = a_0 + r_0, \ldots, v_{t-1} = a_{t-1} + r_{t-1}$, which mean $\tilde{a}_1 = a_1, \ldots, \tilde{a}_{t-1} = a_{t-1}$.
2. **[Secure powers computations]** In this step, for all $\tilde{a}_i$ $(0 \le i \le t-1)$, ciphertexts $E(\tilde{a}_i^2), \ldots, E(\tilde{a}_i^{A-1})$ are generated securely. The following protocol is for $\tilde{a}_i$.
   (a) Each server locally computes $E(\tilde{a}_i) \otimes E(R_i) = E(\tilde{a}_i + R_i \bmod N)$.

(b) By the threshold decryption, a quorum of honest servers cooperatively decrypts $E(\tilde{a}_i + R_i \bmod N)$ to obtain $z_i = \tilde{a}_i + R_i \bmod N$. For any positive integer w, due to $\tilde{a}_i^w = (z_i - R_i)^w \bmod N$, we can express $\tilde{a}_i^w = \alpha_{w,0} + \alpha_{w,1}R_i + \cdots + \alpha_{w,w}R_i^w \bmod N$ by the binomial expansion, for public coefficients $\alpha_{w,0}, \ldots, \alpha_{w,w}$ computed from w, z_i.

(c) This sub-step is repeated for all w $(2 \leq w \leq A)$. Each server locally computes $E(\alpha_{w,0})$ with a common public randomness, and, by using the precomputed $E(R_i), \ldots, E(R_i^w)$, computes $E(\alpha_{w,0}) \otimes E(R_i)^{\alpha_{w,1}} \otimes \cdots \otimes E(R_i^w)^{\alpha_{w,w}} = E(\alpha_{w,0} + \alpha_{w,1}R_i + \cdots \alpha_{w,w}R_i^w) = E(\tilde{a}_i^w)$. After all, the server owns $E(\tilde{a}_i), E(\tilde{a}_i^2), \ldots, E(\tilde{a}_i^A)$.

3. [**Domain check**] This step checks whether $\tilde{a}_i \in [0, A)$ for all $\tilde{a}_i$. The protocol for a_i is as follows:
   (a) As the previous step, by using $E(\tilde{a}_i), E(\tilde{a}_i^2), \ldots, E(\tilde{a}_i^A)$, the server locally computes $E(g(\tilde{a}_i))$.
   (b) A quorum of the servers cooperatively decrypts $E(g(\tilde{a}_i))$ to obtain $g(\tilde{a}_i)$.
   Note that $g(\tilde{a}_i) = 0$ iff $\tilde{a}_i \in [0, A)$. Thus, the servers can verify $\tilde{a}_i \in [0, A)$. Stop if $\tilde{a}_i \notin [0, A)$.
4. [**Bit slicing**] In this step, the final outputs $E(b[0]), \ldots, E(b[k-1])$ are obtained if m is correct, where the following is repeated for all a_i.
   (a) As the above, using precomputed $E(\tilde{a}_i), \ldots, (\tilde{a}_i^{A-1})$, each server locally computes $E(h_u(\tilde{a}_i))$ for all $0 \leq u < \ell_a$. Then, since $E(h_u(\tilde{a}_i)) = E(\tilde{a}_i[u])$ holds due to $\tilde{a}_i \in [0, A)$, the server owns $E(\tilde{a}_i[0]), \ldots, E(\tilde{a}_i[\ell_a - 1])$).
   In the case of correctly computed $E(m)$, since $\tilde{a}_i = a_i$ holds as shown in the section slicing, this sub-step produces $E(a_i[0]), \ldots, E(a_i[\ell_a - 1])$ which mean $E(b[i\ell_a]), \ldots, E(b[(i+1)\ell_a - 1])$. After all, the repeats for all a_i produce $E(b[0]), \ldots, E(b[k-1])$.

Opening. On inputs $E(b[0]), \ldots, E(b[k-1])$ for all the bidders, the servers run the original MPC auction. Finally, the winners and the clearing price are only revealed.

3.4 Security

Full privacy: The ciphertexts, the PKs, the MPC protocols do not reveal the information. In the precomputing phase, since the randomness r_i, R_i are generated by all the honest servers, r_i, R_i are secret unless all the servers are corrupted. In the section slicing step, as long as $E(m)$ is computed correctly, the decrypted value $v_i = \tilde{a}_i + r_i$ does not reveal $\tilde{a}_i$, i.e, a_i, since $|r_i| = \tilde{\ell} > \ell_a + \ell$, where ℓ is sufficiently large. In the secure powers computations step, the decrypted value $z_i = \tilde{a}_i + R_i \bmod N$ does not reveal $\tilde{a}_i$, due to $R_i \in_R \mathbb{Z}_N$. Furthermore, in the domain check step, the decrypted value $g(\tilde{a}_i)$ is always 0 if $E(m)$ is correctly computed, and thus has no information.

Robustness and public verifiability: At first, consider the servers' obstruction. Note that each local computations are verifiable, of course. In the precomputing phase, since only the randomness that the server proved correctly contributes $E(r_i)$ and $E(R_i)$, it is ensured that these ciphertexts includes the secret randomness in the designated interval. Furthermore, the computations of $E(R_i^2 \bmod N), \ldots, E(R_i^A \bmod N)$ are also robust and verifiable, due to the robustness and public verifiability of the MPC multiplication. The conversion and opening phases include the local computations, the MPC protocols and the verifiable threshold decryption, which all supply the robustness and the verifiability.

Next, consider the bidder's obstruction. Assume that $E(m)$ is not correctly computed. For any $m \in \mathbb{Z}_N$, we can express $m = a_0 + a_1\tilde{M} + \cdots + a_t\tilde{M}^t$, where $a_0, \ldots, a_t \in [0, \tilde{M})$, due to $t = \lfloor \ell_N/\ell_{\tilde{M}} \rfloor$. Note that, if $E(m)$ is correctly computed, $a_0, \ldots, a_{t-1} \in [0, A)$ holds. Otherwise, $a_i + r_i > M$ for some i may occur, and thus $\tilde{a}_i \neq a_i$ may holds. If $\tilde{a}_i \notin [0, A)$, it can be detected in the domain check step and gives no obstruction to the bit slicing step. If $\tilde{a}_i \in [0, A)$, $\tilde{a}_i \neq a_i$ may be bit-sliced. Then, honest servers also complete the protocols, though it is possible that the bidder who offers the bad $E(m)$ wins. The clearing price depends on the the randomness r, and it cannot be controlled by the bidder. However, the winner must have the responsibility for the uncontrolled price. Note that an honest bidder can offer the bid that he decides. Furthermore, since the protocol is completed any time, the full privacy is not undermined.

Auction adaptability: This is satisfied, since the original MPC auction can be incorporated with any type of auction.

3.5 Efficiency

It is clear that the bidder's cost is 1 encryption only.

Next, we discuss the server's cost beyond the original MPC auction. The cost depends on the parameters t, ℓ_a (and $A = 2^{\ell_a}$) that are derived from ℓ_n, k. Since the cost additionally depends on the used cryptosystem, we examine the case of the Paillier cryptosystem below. The MPC multiplication, the MPC generation of randomness and the threshold decryption have comparable costs each other, where each server needs some exponentiations and constant round communication among servers. On the other hand, the operation $\otimes$ and the constant multiplication are both the local computations, and the constant multiplication's cost is dominant, which needs an exponentiation. In the precomputing phase, the dominant cost is $2t$ MPC generations of the randomness and tA MPC multiplications per a bid. On the other hand, in the conversion phase, the dominant cost is about $2t$ threshold decryptions per a bid in addition to about $((1/2)A^2t + At + \ell_a At)$ constant multiplications per a bid. Note that the original MPC auction needs k MPC multiplications per a bid, when the circuit for the highest-price auction of [17] is incorporated.

To demonstrate the practicality, we investigate the usual case of $k = 20$. Assume $L \leq 10$. Set security parameters as $\ell_N = 1024, \ell = 128$. Then, other parameters $\ell_{\tilde{M}} = 145$ (i.e., $t = 7$), $\ell_a = 3$ can be set, where ℓ_a is the minimum in this setting. For a bid, the precomputing phase needs about 15 MPC generations of randomness and about 60 MPC multiplications, and the conversion phase needs about 15 threshold decryptions and about 500 constant multiplications, though the original MPC auction needs 20 MPC multiplications. Compared with the original, the precomputing phase is slightly heavy, but it can be executed in advance. On the other hand, in the conversion phase, the cost from the threshold decryptions is comparable to the original. Though the cost from the constant multiplications is heavy, these are local computations and furthermore the computations can be conducted in parallel by powerful servers.

4 Sealed-Bid Auction with Efficient Bids Using Mix-and-Match

In this section, we extend another MPC protocol, called the mix-and-match, into the auction protocol with efficient bids. This extension also adopts the same approach of the MPC conversion as the previous section.

4.1 Building Blocks

For the conversion, we use the Paillier cryptosystem. Note that this system satisfies the properties shown in Section 3.1. On the other hand, the mix-and-match requires another cryptosystem, the ElGamal cryptosystem. The mix-and-match utilizes a mix protocol and a match protocol based on the ElGamal cryptosystem.

ElGamal cryptosystem. Consider a group G with order $|G| = q$ for a large prime q, where $|q| \approx |N|$. G can be constructed as a subgroup of $\mathbb{Z}_p^*$ for a prime p s.t. $q|p-1$, or constructed using elliptic curves. Let g_q be a generator of G. Then, the private key of the cryptosystem is $x \in_R \mathbb{Z}_q$ and the public key is $y = {g_q}^x$. The ElGamal encryption $\tilde{E}(m)$ of m is given by $(g_q^r, y^r m)$, where $r \in_R \mathbb{Z}_q$. Hereafter, we use the notations $E(m)$ for the Paillier encryption and $\tilde{E}(m)$ for the ElGamal encryption. The decryption of the ElGamal ciphertext (C_1, C_2) is made by C_2/C_1^x. Let h_q be another base from G. For $\tilde{E}(h_q^{m_1}) = (g_q^{r_1}, y^{r_1} h_q^{m_1}), \tilde{E}(h_q^{m_2}) = (g_q^{r_2}, y^{r_2} h_q^{m_2})$, define $\tilde{E}(h_q^{m_1}) \otimes \tilde{E}(h_q^{m_2}) = (g_q^{r_1} g_q^{r_2}, y^{r_1} h_q^{m_1} y^{r_2} h_q^{m_2})$, which equals $(g_q^{r_1+r_2}, y^{r_1+r_2} h_q^{m_1+m_2}) = E(h_q^{m_1+m_2})$. Additionally, for $\tilde{E}(h_q^m) = (g_q^r, y^r h_q^m)$, $E(m)^{-1}$ is defined as $((g_q^r)^{-1}, (y^r h_q^m)^{-1})$, which equals $(g_q^{-r}, y^{-r} h_q^{-m})) = E(h_q^{-m})$. Thus, though the ElGamal encryption itself satisfies the multicative homomorphic property, $\tilde{E}(h_q^m)$ on input m satisfies the additive homomorphic property.

This cryptosystem can equip the verifiable threshold decryption [10], where a quorum of servers sharing the private key cooperatively makes the decryption with the public verifiability. This protocol has one round, where each server conducts some exponentiations and a PK, and broadcasts them.

Mix protocol. In the mix protocol, a quorum of servers is given a list $(\alpha_1, \ldots, \alpha_K)$, and outputs a permuted list $(\alpha'_1, \ldots, \alpha'_K)$, where each α_i is a ciphertext or a list of ciphertexts with the size $\tilde{K}$. This protocol ensures that, for a permutation π

on $\{0, \dots, K\}$, α_i and $\alpha_{\pi(i)}$ are derived from the same plaintext or the list of the same plaintexts, but the correspondence of i and $\pi(i)$ is kept secret. This protocol has the public verifiability and robustness. If K is small, Abe's protocol [1] is available, for example, which needs $O(L)$ rounds and server's $O(L\tilde{K}K \log K)$ exponentiations.

Match protocol. The match protocol utilizes the distributed plaintext equality test [14], where, given two ciphertexts $E(m_1), E(m_2)$, a quorum of servers cooperatively checks $m_1 = m_2$ with the secrecy of m_1, m_2, the public verifiability and the robustness. This test protocol has a few round, and each server computes an exponentiation and a PK in addition to a threshold decryption. Then, given a ciphertext $C = E(m)$ and a list of K ciphertexts, the servers cooperatively can pick up $E(m)$ with a different randomness in the list, running the test on input C and every ciphertext in the list. Namely, the match protocol needs K plaintext equality tests.

Mix-and-match. As shown in [14], combining the mix and match protocols, we can obtain an MPC for evaluating a boolean circuit, called the mix-and-match. Therefore, the mix-and-match incorporated with the auction circuit in [17] is, of course, the MPC auction. In this protocol, as inputs of the circuit, a bidder decides the bid price b with k bits, and has to submit $\tilde{E}(h_q^{b[0]}), \dots, \tilde{E}(h_q^{b[k-1]})$ to the servers with the verifiability.

To connect both cryptosystems, we introduce the following tools.

Commitment on $QR(n)$. Let n be an RSA modulus that is the product of safe primes $\tilde{p}, \tilde{q}$ (i.e., $\tilde{p} = 2\tilde{p}'+1$, $\tilde{q} = 2\tilde{q}'+1$, where $\tilde{p}', \tilde{q}'$ are primes), and let $QR(n)$ be a subgroup of quadratic residues in $\mathbb{Z}_n^*$. Under strong RSA assumption [11], the following commitment scheme is available [5, 18]. Let $g_n, h_n \in_R QR(n)$. Then, a sender who does not know the factorization of n can make a commitment of $m \in \mathbb{Z}$ as $Com(m) = g_n^m h_n^r$, where $r \in_R \mathbb{Z}_n$.

PK between Paillier and ElGamal cryptosystems. In [9], a PK proving the knowledge m s.t. $C_1 = E(m)$ and $C_2 = Com(m)$ is described. Concretly, this PK proves the knowledge m, r_1, r_2 s.t. $C_1 = g^m r_1^N \bmod N^2$ and $C_2 = g_n^m h_n^{r_2}$. On the other hand, in [6], a PK proving the discrete logs in the different groups including $QR(n)$ is presented, and thus this PK can show the knowledge m s.t. $C_2 = Com(m)$ and $(C_3, C_4) = \tilde{E}(h_q^m)$. Concretly, this PK proves the knowledge m, r_2, r_3 s.t. $C_2 = g_n^m h_n^{r_2}$ and $C_3 = g_q^{r_3}, C_4 = y^{r_3} h_q^m$. Therefore, combining two PKs, we can obtain PK proving the knowledge m s.t. $C_1 = E(m)$ and $(C_3, C_4) = \tilde{E}(h_q^m)$. The cost of prover or verifier is less than 10 multi-exponentiations.

Note that it is necessary that the factorization of n is kept secret for the strong RSA assumption. The secure computation of n with safe primes without revealing the factorization can be archived by an efficient MPC in [3].

4.2 Proposed Protocols

The overview is the similar to the protocols in the previous section. Though the bidder submits a Paillier ciphertext, this is converted into ElGamal ciphertexts of bits of his bid, using the mix and match protocols.

Setup. This is the same as the setup in Section 3.3, except for the setup of polynomials $g(x), h_u(x)$, which is not necessary.

Precomputing.

1. The servers cooperatively generate a Paillier ciphertext $E(r_i)$, and additionally an ElGamal ciphertext $\tilde{E}(h_q^{r_i})$ with $0 \leq i \leq t-1$, where r_i is a uniformly random plaintext on interval $[0, 2^{\tilde{\ell}})$ that is contributed by all honest servers. Concretely, each S_j conducts the following:
   (a) S_j generates $E(r_{ij})$ and $\tilde{E}(h_q^{r_{ij}})$, where $r_{ij} \in_R [0, 2^{\ell_a+\ell})$.
   (b) For the ciphertexts $E(r_{ij})$ and $\tilde{E}(h_q^{r_{ij}})$, S_j proves the interval $r_{ij} \in [0, 2^{\ell_a+\ell})$ and that $E(r_{ij})$ and $\tilde{E}(h_q^{r_{ij}})$ have the same r_{ij}.

   After the above sub-steps, let Q be the set of indices of the servers proved the interval correctly. Without loss of generality, let $Q = \{1, \ldots, |Q|\}$. Then, each honest server sets $E(r_i) = E(r_{i1}) \otimes \cdots \otimes E(r_{i|Q|})$ and $\tilde{E}(h_q^{r_i}) = \tilde{E}(h_q^{r_{i1}}) \otimes \cdots \otimes \tilde{E}(h_q^{r_{i|Q|}})$. Note that $E(r_i) = E(\Sigma_{j \in Q} r_{ij})$ and $\tilde{E}(h_q^{r_i}) = \tilde{E}(h_q^{\Sigma_{j \in Q} r_{ij}})$. Furthermore, each server locally computes $E(r) = E(r_0) \otimes E(r_1)^{\tilde{M}} \otimes \cdots \otimes E(r_{t-1})^{\tilde{M}^{t-1}}$.
2. As the preparation for the bit slicing with the domain check below, the servers perform the following for all $0 \leq i \leq t-1$.
   (a) Consider the list $\lambda_i = (\epsilon_0, \ldots, \epsilon_{A-1})$, where ϵ_u $(0 \leq u \leq A-1)$ is a list $(\tilde{E}(h_q^u), \tilde{E}(h_q^{u[0]}), \ldots, \tilde{E}(h_q^{u[\ell_a-1]}))$. Each server locally computes λ_i, where the randomness of each ciphertext is publicly fixed as 0.
   (b) A quorum of the servers cooperatively performs the mix protocol on input λ_i, and outputs a permuted list $\tilde{\lambda}_i$.

Bidding. As well as Section 3.3, a bidder decides his bid b with k bits, and submits $E(m) = E(f(b))$ to the servers.

Conversion. Here, we show the protocol for a single bid, which should be performed for every bid that needs the conversion. The common input of the servers is $E(m) = E(f(b))$. It is converted to $\tilde{E}(h_q^{b[0]}), \ldots, \tilde{E}(h_q^{b[k-1]})$, as follows:

1. **[Section slicing]** The sub-steps $(a), (b), (c)$ are exactly the same as those in Section 3.3. As a result, the servers obtain $v = m + r \bmod N$, and randomized sections $v_0 = (v[\ell_{\tilde{M}}-1], \ldots, v[0])_2, \ldots, v_{t-1} = (v[t\ell_{\tilde{M}}-1], \ldots, v[(t-1)\ell_{\tilde{M}}])_2$ by $\ell_{\tilde{M}}$ bits. As the previous, define $\tilde{a}_0 = v_0 - r_0 \bmod q, \ldots, \tilde{a}_{t-1} = v_{t-1} - r_{t-1} \bmod q$. Then, as the step (c), the servers perform the following:
   (c) Each server locally computes $\tilde{E}(h_q^{v_i})$ with a common public randomness, and computes $\tilde{E}(h_q^{v_i}) \otimes \tilde{E}(h_q^{r_i})^{-1} = \tilde{E}(h_q^{v_i - r_i \bmod q}) = \tilde{E}(h_q^{\tilde{a}_i})$ for all i.

From the similar discussion to Section 3.3, $\tilde{a}_1 = a_1, \ldots, \tilde{a}_{t-1} = a_{t-1}$ hold, if $E(m)$ is correctly computed.

2. **[Bit slicing with domain check]** Using the mix-and-match, the servers perform the domain check and the bit slicing simultaneously.
   (a) Given $\tilde{E}(h_q^{\tilde{a}_i})$ and the mixed list $\tilde{\lambda}_i$, a quorum of servers cooperatively performs the match protocol to pick up $(\tilde{E}(h_q^a), \tilde{E}(h_q^{a[0]}), \ldots, \tilde{E}(h_q^{a[\ell_a-1]}))$ with $\tilde{a}_i = a$, where the first element $\tilde{E}(h_q^a)$ from each list in $\tilde{\lambda}_i$ is compared with $\tilde{E}(h_q^{\tilde{a}_i})$ by the plaintext equality test protocol. If $\tilde{a}_i \in [0, A)$, the match is found. As mentioned above, note that, if $E(m)$ is correctly computed, $\tilde{a}_i = a_i$ holds. Then, the servers obtain $\tilde{E}(h_q^{a[0]}) = \tilde{E}(h_q^{\tilde{a}_i[0]}) = \tilde{E}(h_q^{a_i[0]}) = \tilde{E}(h_q^{b[i\ell_a]}), \ldots, \tilde{E}(h_q^{a[\ell_a-1]}) = \tilde{E}(h_q^{\tilde{a}_i[\ell_a-1]}) = \tilde{E}(h_q^{a_i[\ell_a-1]}) = \tilde{E}(h_q^{b[(i+1)\ell_a-1]})$. If no match, i.e., $\tilde{a}_i \notin [0, A)$, then stop the protocol.

   After the repetition of all $\tilde{a}_i$, the servers obtain $\tilde{E}(h_q^{b[0]}), \ldots, \tilde{E}(h_q^{b[k-1]})$.

Opening. On inputs $\tilde{E}(h_q^{b[0]}), \ldots, \tilde{E}(h_q^{b[k-1]})$ for all the bidders, the servers run the mix-and-match for the auction circuit [17]. Finally, the winners and the clearing price are only revealed.

4.3 Security

Full privacy: As well as the protocols in the previous section, the ciphertexts, the PKs, the MPC protocols, and the randomized sections v_i do not reveal the information. The difference is the bit-slicing with domain check by using the mix and match protocols. Due to the mix protocol on $\tilde{\lambda}_i$ in the precomputing phase, the matched $(\tilde{E}(h_q^a), \tilde{E}(h_q^{a[0]}), \ldots, \tilde{E}(h_q^{a[\ell_a-1]}))$ from $\tilde{\lambda}_i$ reveals no information on $a = \tilde{a}_i$. Therefore, the full privacy holds.

Robustness and public verifiability: The server's obstruction is protected, since the mix and match protocols provide the robustness and public verifiability, in addition to the discussion of these properties in the previous section. These properties against the bidder's obstruction are shown by the similar discussion to the previous section.

Auction adaptability: This is satisfied, since the original mix-and-match can be incorporated with any type of auction.

4.4 Efficiency

It is clear that the bidder's cost is 1 encryption only. However, note that the bidder is required a Paillier encryption, instead of ElGamal encryptions in the original mix-and-match auction. As discussed in [21], the computation cost of a single Paillier encryption is approximately that of 3 ElGamal encryptions, when the message space has the same size. Thus, in the usual case of $k = 20$, the bidder's encryption cost is about $3/20 \approx 1/7$ of the original auction which forces a bidder to compute 20 ElGamal encryptions. In addition, in our protocols, it is not necessary that the bidder conducts any zero-knowledge proof.

Next, we discuss the server's cost beyond the original mix-and-match auction. The cost depends on the parameters t, ℓ_a (and $A = 2^{\ell_a}$). In the precomputing phase, the dominant cost is t MPC generations of the randomness and t mix protocols. Each mix protocol is performed on inputs $(\ell_a+1)A$ ciphertexts. On the other hand, in the conversion phase, the dominant cost is t match protocols per a bid. Each match protocol needs A plaintext equality tests. Note that the original mix-and-match auction needs k mix and match protocols per a bid, when the circuit for the highest-price auction of [17] is incorporated. Each mix protocol is performed on 12 ciphertexts, and each match protocol needs 8 plaintext equality tests.

As the previous section, to demonstrate the practicality, we investigate the case of $k = 20$, $t = 7$ and $\ell_a = 3$. For a bid, the precomputing phase needs 7 MPC generations of randomness and 7 mix protocols on 32 ciphertexts, and the conversion phase needs 7 match protocols on 8 plaintext equality tests. On the other hand, the original mix-and-match auction needs 20 mix protocols on 12 ciphertexts and 20 match protocols on 8 plaintext equality tests. The heavy precomputing phase can be conducted before the auction, and the cost of the conversion phase is less than the cost of the original mix-and-match auction.

5 Conclusion

This paper has proposed sealed-bid auction protocols, where a bidder can submit only single ciphertext on his bid. The novel core technique is the MPC to convert from the single ciphertext to ciphertexts on each bit of his bid, which are followed by the efficient MPC auctions.

A future work is to formalize the security of the MPC of the conversion, and to prove it. Another is an exploration of other applications where the conversion is effective.

References

[1] M. Abe, "Mix-networks on permutation networks," Advances in Cryptology — ASIACRYPT '99, LNCS 1716, pp.258–273, Springer–Verlag, 1999. 240

[2] M. Abe and K. Suzuki, "$m + 1$-st price auction using homomorphic encryption," Proc. 4th International Workshop on Practice and Theory in Public Key Cryptography (PKC 2002), LNCS 2274, pp.115–124, Springer–Verlag, 2002. 231

[3] J. Algesheimer, J. Camenisch, and V. Shoup, "Efficient computation modulo a shared secret with application to the generation of shared safe-prime products," Advances in Cryptology — CRYPTO 2002, LNCS 2442, pp.417–432, Springer–Verlag, 2002. 240

[4] C. Cachin, "Efficient private bidding and auctions with an oblivious third party," Proc. 6th ACM Conference on Computer and Communications Security (ACM–CCS '99), pp.120–127, 1999. 231

[5] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," Proc. Third Conference on Security in Communication Networks (SCN'02), LNCS 2576, Springer–Verlag, 2002. 240

[6] J. Camenisch and M. Michels, "Separability and efficiency for generic group signature schemes," Advances in Cryptology — CRYPTO '99, LNCS1666, pp.413–430, Springer–Verlag, 1999. 240

[7] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," Advances in Cryptography — EUROCRYPT 2001, LNCS 2045, pp.280–300, Springer–Verlag, 2001. 231, 232, 233, 234

[8] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," Proc. 4th International Workshop on Practice and Theory in Public Key Cryptography (PKC 2001), pp.119–136, Springer–Verlag, 2001. 234

[9] I. Damgård and M. Jurik, "Client/server tradeoffs for online elections," Proc. 5th International Workshop on Practice and Theory in Public Key Cryptography (PKC 2002), pp.125–140, Springer–Verlag, 2002. 234, 235, 240

[10] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," Advances in Cryptology — CRYPTO'89, LNCS 435, pp.307–315, Springer–Verlag, 1990. 239

[11] E. Fujisaki and T. Okamoto, "Statistical zero knowledge protocols to prove modular polynomial relations," Advances in Cryptology — CRYPTO '97, LNCS 1294, pp.16–30, Springer–Verlag, 1997. 240

[12] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," Proc. 19th Annual ACM Symposium on Theory of Computing (STOC '87), pp.218–229, 1987. 231

[13] M. Harkavy, J. D. Tygar, and H. Kikuchi, "Electronic auctions with private bids," Proc. 3rd USENIX Workshop on Electronic Commerce, pp.61–74, 1998. 230, 231

[14] M. Jakobsson and A. Juels, "Mix and match: Secure function evaluation via ciphertexts," Advances in Cryptography — ASIACRYPT2000, LNCS 1976, pp.162–177, Springer–Verlag, 2000. 231, 232, 240

[15] A. Juels and M. Szydlo, "A two-server, sealed-bid auction protocol," Proc. 6th Financial Cryptography Conference (FC 2002), LNCS 2357, pp.72–86, Springer–Verlag, 2003. 232

[16] H. Kikuchi, "$(m+1)$st-price auction protocol," Proc. 5th Financial Cryptography Conference (FC 2001), LNCS 2339, pp.351–363, Springer–Verlag, 2002. 230, 231

[17] K. Kurosawa and W. Ogata, "Bit-slice auction circuit," Proc. 7th European Symposium on Research in Computer Security (ESORICS 2002), LNCS 2502, pp.24–38, Springer–Verlag, 2002. 231, 234, 238, 240, 242, 243

[18] A. Lysyanskaya, Signature Schemes and Applications to Cryptographic Protocol Design, Ph.D. thesis, Massachusetts Institute of Technology, 2002. Available in `http://www.cs.brown.edu/~anna/phd.ps`. 240

[19] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," Proc. 1st ACM Conference on Electronic Commerce, pp.129–139, ACM Press, 1999. 232

[20] K. Omote and A. Miyaji, "A second-price sealed-bid auction with verifiable discriminant of p_0-th root," Proc. 6th Financial Cryptography Conference (FC 2002), LNCS 2357, pp.57–71, Springer–Verlag, 2003. 231

[21] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," Advances in Cryptology — EUROCRYPT '99, pp.223–238, Springer–Verlag, 1999. 231, 234, 242

Providing Receipt-Freeness in Mixnet-Based Voting Protocols

Byoungcheon Lee[1,2], Colin Boyd[1], Ed Dawson[1], Kwangjo Kim[3], Jeongmo Yang[2], and Seungjae Yoo[2]

[1] Information Security Research Center,
Queensland University of Technology,
GPO Box 2434, Brisbane, QLD 4001, Australia,
{b6.lee,c.boyd,e.dawson}@qut.edu.au
[2] Joongbu University,
101 Daehak-Ro, Chuboo-Meon, Kumsan-Gun, Chungnam, 312-702, Korea,
{sultan,jmyang,sjyoo}@joongbu.ac.kr
[3] Information and Communications University,
58-4, Hwaam-dong, Yusong-gu, Daejeon, 305-732, Korea,
kkj@icu.ac.kr

Abstract. It had been thought that it is difficult to provide receipt-freeness in mixnet-based electronic voting schemes. Any kind of user chosen randomness can be used to construct a receipt, since a user can prove to a buyer how he had encrypted the ballot. In this paper we propose a simple and efficient method to incorporate receipt-freeness in mixnet-based electronic voting schemes by using the well known re-encryption technique and designated verifier re-encryption proof (DVRP). In our scheme a voter has to prepare his encrypted ballot through a randomization service provided by a tamper resistant randomizer (TRR), in such a way that he finally loses his knowledge on randomness. This method can be used in most mixnet-based electronic voting scheme to provide receipt-freeness.

Keywords. Electronic voting, Receipt-freeness, Mixnet, Re-encryption, Designated-verifier re-encryption proof, Tamper resistant randomizer.

1 Introduction

With the research on electronic voting we try to implement real world voting procedure in electronic means using computer and network. Since we are already accustomed to many electronic means such as high speed computer, Internet, as well as mobile communications, we also expect that voting would be provided through electronic means. It seems to be a worldwide trend that the participation rate in election is decreasing, specially younger generations do not want to participate in voting if they are not interested in. Electronic voting can be a good solution against this problem by providing easier and friendlier means of voting.

Over the world many kind of electoral systems are currently being used [ES]. Those electoral systems can be classified into the following two categories; plurality systems and majoritorian systems. First, the plurality system is relatively simple; the candidate who receives the most votes is elected regardless of whether the candidate receives a majority of votes. Second, the majoritorian system is a little more complicated; a winning candidate is required to receive an absolute majority (more than half) of the vote. In Australian alternative voting (preferential voting) system, voter's next preferences are used to decide a majority winner in later rounds, if one does not emerge from the first round of counting.

In cryptographic research electronic voting is one of the most challenging cryptographic protocol problems, because it has extensive security requirements to be satisfied, and some of them are quite contradictory.

Security requirements.

– **Privacy**: Ensures the secrecy of the contents of ballots. Usually it is achieved by encrypting the ballot with the public key of a group of authorities and by trusting them.
– **Prevention of double voting**: Ensures that any eligible voters can vote only once. To achieve this the authority needs to check the authenticity and eligibility of the voter and record the participation in his secure database. If a public bulletin board is used as a public communication channel (most messages are posted there) and voters access it in an authenticated manner, double voting can be prevented easily.
– **Universal verifiability**: Ensures that any party can be convinced that all valid votes have been included in the final tally. To achieve this all the relevant messages need to be published and the correctness of all processes (voting, mixing, tally) should be publicly verifiable.
– **Fairness**: Ensures that no partial tally is revealed to anyone before the end of the election procedure, as it may affect the voting result in some way. Sometimes we need to trust that the authorities will not reveal partial tally.
– **Robustness**: Ensures that the voting system can tolerate a certain number of faulty participants.
– **Receipt-freeness**: Ensures that a voter neither obtains nor is able to construct a receipt which can prove the content of his vote. To achieve this the voter should not be able to choose his own randomness when he prepares his ballot. A user chosen randomness can work as a receipt.

Sometimes vote buying and coercion issues are discussed as an important security requirement. But, without receipt voters cannot be actively involved in vote buying or coercion. Without receipt these issues would be limited to social problems and are out of the scope of our discussion.

Main Approaches. Electronic voting schemes found in the literature can be classified by their approaches into the following three categories:

– Schemes using blind signature: [Cha88], [FOO92], [Ohk99], [Kim01].
– Schemes using mix-net: [PIK93], [SK95], [Abe98], [Jak98a], [Jak98b], [Abe99], [HS00], [FS01], [Nef01], [Gol02], [Gro03].
– Schemes using homomorphic encryption: [Ben87], [SK94], [CFSY96], [CGS97], [LK00], [HS00], [Hirt01], [Bau01], [Hirt01], [MBC01], [LK02].

Voting schemes based on blind signature are simple, efficient, and flexible, but require the existence of anonymous channel. Frequently, an anonymous channel is implemented using mixnet, but if a secure mixnet is available, a blind signature is not required anymore. Voting schemes based on mixnet are generally not efficient because they require huge amounts of computation for multiple mixers to prove the correctness of their mixing. But recent results of [FS01], [Nef01], [Gro03], and [Gol02] have greatly improved the efficiency of mixnet. Voting schemes based on homomorphic encryption are efficient in the opening stage, but intensive zero-knowledge proofs are used to prove the validity of each ballot in the voting stage, which are costly for the voters. Note that much extensive research on receipt-freeness had been done in this approach.

Our Contribution. In this paper we propose a simple and efficient method to incorporate receipt-freeness in mixnet-based electronic voting schemes by using the well known re-encryption technique and designated-verifier re-encryption proof (DVRP). In our scheme a voter has to prepare his final encrypted ballot through a randomization service provided by a tamper resistant randomizer (TRR), in such a way that he finally loses his knowledge on randomness. This method can be used in most mixnet-based electronic voting schemes to provide receipt-freeness.

Outline of the Paper. The rest of the paper is organized as follows. In Section 2 we review several background concepts and related works in electronic voting. In Section 3 we describe our voting model such as entities and their roles, communication model, and assumptions. In Section 4 we describe some cryptographic primitives such as threshold ElGamal encryption and designated-verifier re-encryption proof, which are required to describe the proposed protocol. The proposed voting protocol is described in Section 5 and is analyzed in Section 6. Finally, we conclude and discuss future works in Section 7.

2 Background and Related Works

2.1 Receipt-Freeness

Receipt-freeness is a unique security requirement of electronic voting systems which distinguishes it from other cryptographic protocol problems. Also it is a very important property in electronic voting, since vote buying and coercion are very common experiences in real world election scenarios. It is thought that without providing receipt-freeness electronic voting schemes cannot be used for real political elections.

The concept of receipt-freeness was first introduced by Benaloh and Tuinstra [BT94]. Considering the threat of vote buyers (or coercers), a voting scheme should ensure not only that a voter *can* keep his vote private, but also that he *must* keep it private. The voter should not be able to prove to a third party that he had cast a particular vote. He must neither obtain nor be able to construct a receipt which can prove the content of his vote.

To achieve receipt-freeness, voting schemes in the literature use some kind of trusted authority which provide a randomization service and make some physical assumption about the communication channel between the voter and the authority depending on the design of the protocol.

1. One-way untappable channel from the voter to the authority [Oka97].
2. One-way untappable channel from the authority to the voter [SK95, HS00].
3. Two-way untappable channel (voting booth) between the voter and the authority [BT94, Hirt01, LK02].

Note that research on receipt-freeness had been done mainly in homomorphic encryption based voting schemes, since designing a receipt-free scheme is relatively easy in those schemes by using zero-knowledge proof techniques. On the other hand, in blind signature based or mixnet-based schemes, voter chosen randomness can be used as a receipt. The voter can prove the content of his encrypted ballot using his knowledge of randomness.

[LK00] tried to provide receipt-freeness by extending [CGS97]. They assumed a trusted third party called honest verifier (HV) who verifies the validity of voter's first ballot and provides a randomization service, *i.e.*, HV re-encrypts it to generate the final ballot and generates the proof of validity of ballot cooperatively with the voter such that the voter cannot obtain any receipt. But [Hirt01] has pointed out that in this protocol a malicious HV can help a voter to cast an invalid vote and thereby falsify the outcome of the whole vote. Moreover the voter can construct a receipt by choosing his challenge as a hash value of his first ballot. This is the same attack applied to [BT94]. To resist against this attack, voter should not be allowed to choose any challenge. [Hirt01] fixed [LK00] and proposed a receipt-free voting scheme based on a third-party randomizer. The role of the randomizer is similar to HV of [LK00], but the randomizer generates the re-encryption proof in a designated-verifier manner and generates the proof of validity using a divertible zero-knowledge proof technique.

[HS00] provided receipt-freeness in homomorphic encryption based voting requiring only one way untappable channel from a randomizer to voters with a cost of huge computation of the randomizer. In this scheme the randomizer works as a kind of personal mixer who presents randomized ballots to a voter in a designated-verifier manner.

[MBC01] proposed a receipt-free electronic voting protocol using a tamper-resistant smartcard which plays the role of personal mixer. But in their voting protocol the re-encryption proof is given in an interactive manner, therefore the same attack applied to [BT94] and [LK00] is possible. [LK02] fixed [MBC01] and proposed a receipt-free electronic voting scheme in which a tamper-resistant randomizer (TRR) replaces the role of untappable channel and a third party randomizer. In this scheme voter prepares an encrypted ballot through an interactive protocol with TRR in a way that he loses his randomness but is convinced personally that the final ballot is constructed correctly.

All of these previous works are homomorphic encryption based voting schemes. In this paper we suggest a simple and efficient method to incorporate receipt-freeness in mixnet-based voting schemes by using similar randomization technique. In mixnet-based voting schemes this kind of randomization technique can be applied more efficiently since we do not need to prove the validity of each ballot. To the best of our knowledge, this is the first work to incorporate receipt-freeness in mixnet-based voting schemes.

2.2 Mixnet-Based Voting Schemes

The notion of a mixnet was first introduced by Chaum [Cha88], and further developed by a number of researchers. A mixnet enables a set of senders to send their messages anonymously, thus it is a primitive to provide anonymity service. Mixnets can be classified into decryption mixnet and re-encryption mixnet depending on mixing mechanism. The original proposal of Chaum was a decryption mixnet, but many recent works deal with re-encryption mixnet, since it can separate mixing and decryption phases, which provides more flexibility, robustness, and efficiency. Mixnets can also be classified into verifiable mixnet and optimistic mixnet depending on correctness proof.

In verifiable mixnet each server provides proofs that its shuffling is correct, thus the correctness of mixing is publicly verifiable. Abe [Abe99] proposed a general framework of permutation mixnets. Recent works by [FS01], [Nef01], and [Gro03] have shown breakthrough progress in the construction of verifiable mixnet and the proving technique of a correct shuffle. [FS01] represented a shuffling as a matrix between inputs and outputs and proved that it is a correct permutation. [Nef01] has provided verifiable permutation using an iterated logarithmic multiplication proof. [Gro03] used a homomorphic multicommitment scheme to provide a more efficient shuffling proof.

On the other hand, in optimistic mixnet the verification of correct shuffling is not provided by each server. Instead, the correctness of the shuffling of the whole mixnet is verified after the mixnet outputs the shuffling results in plaintexts. Drawbacks of optimistic mixnets include that a cheating server cannot be identified instantly and some outputs are revealed in plaintexts even when the shuffling is incorrect. Jakobsson tried to design efficient optimistic mixnets [Jak98a, Jak98b]. More recently Golle *et. al.* [Gol02] has shown more efficient mixnet by using an optimistic approach, *i.e.*, they proved only the preservation of the product of messages after mixing, not proving the correctness of mixing of each message. But there exists some criticisms [Wik02, AI03] to this approach.

2.3 Tamper-Resistant Hardware Device

To provide receipt-freeness we need to introduce a trusted third party randomizer and untappable channel between voters and the randomizer. But, in the real world, implementing an untappable channel in distributed environments is very difficult. If a physically isolated voting booth in a dedicated computer network is used to achieve untappable channel, it will be expensive and inconvenient for voters since they have to go to a particular voting booth. Also, assuming a trusted third party randomizer is a burden.

As suggested in [MBC01] and [LK02], a tamper-resistant hardware device can replace the role of untappable channel and a trusted third party. Moreover, a tamper-resistant hardware device is thought to be the ultimate place to store user's secret information such as secret signing key, since it is designed by secure architecture and has limited interface. We expect that it will be available to most users in the near future. In this paper we use a hardware device called tamper resistant randomizer (TRR) to provide a randomization service to voter's encrypted ballot.

3 Voting Model

Overview. A typical mixnet-based electronic voting scheme runs as follows.

1. Voting: A voter prepares an encrypted ballot and posts it on a bulletin board in an authenticated manner with his signature.
2. Mixing: Multiple independent mix servers shuffle the posted ballots sequentially in a verifiable way such that the voter-vote relationship is lost.
3. Tally: After the mixing process is finished, multiple tally servers jointly open encrypted ballots using threshold decryption protocol.

Our main idea is quite simple. We assume a third party randomizer which provides randomization service; it receives the voter's first ballot, randomizes it by using re-encryption to generate a final ballot, and gives it to the voter in an authenticated way. A voter is required to make his final encrypted ballot through an interactive protocol with the randomizer. In this paper the randomizer is implemented by a secure hardware device called tamper resistant randomizer (TRR). In the randomization process, the randomizer provides a designated-verifier re-encryption proof (DVRP) to the voter, so the voter is convinced personally that his final ballot is constructed correctly, but he cannot transfer the proof to others. Through the randomization process the voter loses his knowledge of randomness, thus he cannot construct a receipt.

Entities and Their Roles. Main entities involved in the proposed voting protocol are an administrator A, l voters V_i ($i = \{1, \ldots, l\}$), m mixers M_j ($j = \{1, \ldots, m\}$), and n talliers T_k ($k = \{1, \ldots, n\}$). The roles of each entity are as follows:

– Administrator A manages the whole voting process. A announces the list of candidates, the list of eligible voters, and system parameters including the public key for ballot encryption. A issues TRRs to eligible voters in the registration stage. A publishes the voting result.

– Voter V_i participates in voting. We assume that a voter is certified properly, for example, using PKI, and has a signing key corresponding to the certified public key. V_i needs to identify and register himself to A, then A issues a TRR_i to him which is securely equipped with its own signing key. In the voting stage V_i generates a final encrypted ballot through an interactive protocol with TRR_i and posts it on the bulletin board with his signature.
– Mixers M_j provide mixing service for the collected ballots such that the voter-vote relationship is lost.
– Talliers T_k share the private key of the voting scheme in a (t, n)-threshold verifiable secret sharing (VSS) scheme. After the mixing stage is finished, they cooperatively open each ballot using the (t, n)-threshold decryption protocol.

Tamper resistant randomizer. A tamper resistant randomizer (TRR) is a secure hardware device owned by a voter which works in voter's computer system. It is securely equipped with its own signing key and voter's certified public key. It has the functionality of computing re-encryption and designated-verifier re-encryption proof (DVRP). The communication channel between voter and TRR is an internal channel which does not use network functionality. It is assumed that any party over the network cannot observe the internal communication. It provides a randomization service to voter's encrypted ballot; receives voter's first ballot, re-encrypts it to generate a final ballot, and gives it to the voter in an authenticated manner using its signature. It also provides DVRP to the voter.

Communication Model. In this paper a bulletin board is used as a public communication channel with memory. It can be read by anyone, but only legitimate parties can write messages on it in an authenticated manner. Once a message is written on bulletin board, it cannot be deleted or overwritten. Most messages and proofs of the protocol will be posted on the bulletin board. It is a main communication tool to provide the voting protocol with a universal verifiability. For the voting system to be reliable the bulletin board system should be tamper-proof and resistant against the denial of service (DoS) attack.

The communication channel between involved parties (voters, mixers, talliers) and bulletin board is a public channel such as the Internet. But the communication between a voter and his TRR is an internal channel which cannot be observed by a buyer. We need to assume that a buyer cannot observe the very moment of voter's voting. This is a basic assumption to get receipt-freeness.

Ballot encoding. In the proposed voting protocol any fixed encoding format can be used like most mixnet-based voting schemes, possibly within the size of a modular number. Therefore the proposed protocol provides extensive flexibility and can be used for a wide range of complicated real world voting schemes, for example, Australian preferential voting. Note that, if the encoding format is not fixed, there is a possibility that a specific encoding agreed between a voter and a buyer can be used as a receipt.

4 Cryptographic Primitives

In this paper ballots are encrypted with ElGamal encryption which will be decrypted through a (t, n)-threshold decryption protocol and re-encryption mixnet is used to provide anonymity service.

Consider the ElGamal encryption scheme [ElG85] under a multiplicative subgroup Z_p^* of order q, where p and q are large primes such that $q \mid p-1$. If a receiver chooses a private key s, the corresponding public key is $h = g^s$ where g is the generator of the subgroup. Given a message $m \in \langle g \rangle$, encryption of m is given by $(x, y) = (g^\alpha, h^\alpha m)$ for a randomly chosen $\alpha \in_R Z_q$. To decrypt the ciphertext (x, y), the receiver recovers the plaintext as $m = y/x^s$ using the private key s.

4.1 Re-encryption and Mixnet

ElGamal is a probabilistic encryption scheme that allows re-randomization of ciphertexts. Given an ElGamal ciphertext (x, y), a mix server can efficiently compute a new ciphertext $(x', y') = (xg^r, yh^r)$, choosing $r \in_R Z_q^*$ at random, that decrypts to the same plaintext as (x, y). This re-encryption can be computed by anyone (it does not require the knowledge of the private key) and the exponentiations can be pre-computed.

Given two ElGamal ciphertexts, it is infeasible to determine whether one is a re-encryption of the other without knowledge of either the private key s or the re-encryption factor r, assuming that the Decisional Diffie-Hellman problem is hard in Z_q. Using this property a mix server can hide the correspondence between its input and output ciphertexts, while preserving messages, by outputing re-encrypted ciphertexts in a random order.

4.2 Designated-Verifier Re-encryption Proofs

A designated-verifier proof is a proof which is convincing only to the designated verifier, but it is completely useless when transferred to any other entity [JSI96].

The basic idea is to prove the knowledge of either the witness in question or the secret key of the designated verifier. Such a proof convinces the designated verifier personally because he assumes that the prover does not know his secret key. But, if the proof is transferred to another entity, it loses its persuasiveness completely.

We consider designated-verifier re-encryption proofs (DVRP). Let $(x, y) = (g^\alpha, h^\alpha m)$ be an original ElGamal ciphertext of some message m with a public key $h = g^s$. Let $(x_f, y_f) = (xg^\beta, yh^\beta)$ be a re-encrypted ElGamal ciphertext generated by the prover P (TRR). Let $h_V = g^{s_V}$ be the public key of the verifier V (Voter) corresponding to the private key s_V. P wants to prove to V that his re-encryption was generated correctly in a way that his proof cannot be transferred to others. He will prove that x_f/x and y_f/y have the same discrete logarithm β under bases g and h, respectively.

Designated-verifier re-encryption proof (DVRP):

Prover (TRR):

1. Chooses $k, r, t \in_R Z_q$.
2. Computes $(a, b) = (g^k, h^k)$ and $d = g^r h_V^t$.
3. Computes $c = H(a, b, d, x_f, y_f)$ and $u = k - \beta(c + r)$.
4. Sends (c, r, t, u) to V.

Verifier (Voter):

1. Verifies $c \overset{?}{=} H(g^u (x_f/x)^{c+r}, h^u (y_f/y)^{c+r}, g^r h_V^t, x_f, y_f)$.

In this protocol $d = g^r h_V^t$ is a trapdoor commitment (or chameleon commitment) for r and t. Because V knows his private key s_V, he can open d to arbitrary values r' and t' such that $r' + s_V t' = r + s_V t$ holds. V can generate the re-encryption proof for any $(\tilde{x}, \tilde{y})$ of his choice using his knowledge of s_V. Selecting $(\tilde{\gamma}, \tilde{\delta}, \tilde{u})$ at random, V computes

$$\tilde{c} = H(g^{\tilde{u}} (x_f/\tilde{x})^{\tilde{\gamma}}, h^{\tilde{u}} (y_f/\tilde{y})^{\tilde{\gamma}}, g^{\tilde{\delta}}, x_f, y_f),$$

and also computes $\tilde{r} = \tilde{\gamma} - \tilde{c}$ and $\tilde{t} = (\tilde{\delta} - \tilde{r})/s_V$. Then $(\tilde{c}, \tilde{r}, \tilde{t}, \tilde{u})$ is an accepting proof. Therefore designated-verifier re-encryption proof cannot be transferred to others.

4.3 Threshold ElGamal Encryption

A threshold public-key encryption scheme is used to share a secret key among n talliers such that messages can be decrypted only when a substantial subset of talliers cooperate. More detailed description is found in [CGS97] and [Ped91]. It consists of key generation protocol, encryption algorithm, and decryption protocol.

Consider a (t, n)-threshold decryption scheme where the secret key is shared among n talliers T_k $(1 \leq k \leq n)$ and decryption is possible only when more than t talliers cooperate. Through the key generation protocol, each tallier T_k will possess a share $s_k \in Z_q$ of a secret s. Each tallier publishes the value $h_k = g^{s_k}$ as a commitment of the share s_k. The shares s_k are chosen such that the secret s can be reconstructed from any subset Λ of t shares using the appropriate Lagrange coefficients,

$$s = \sum_{k \in \Lambda} s_k \lambda_{k,\Lambda}, \quad \lambda_{k,\Lambda} = \prod_{l \in \Lambda \setminus \{k\}} \frac{l}{l - k}.$$

The public key $h = g^s$ is published to all participants in the system.

Encryption of a message m using the public key h is given by $(x, y) = (g^\alpha, h^\alpha m)$ which is the same as the ordinary ElGamal encryption. To decrypt a ciphertext $(x, y) = (g^\alpha, h^\alpha m)$ without reconstructing the secret s, talliers execute the following protocol:

1. Each tallier T_k broadcasts $w_k = x^{s_k}$ and proves the equality of the following discrete logs in zero-knowledge using the proof of knowledge protocol,
$$\log_g h_k = \log_x w_k.$$
2. Let Λ denote any subset of talliers who passed the zero-knowledge proof. Then the plaintext can be recovered as
$$m = y / \prod_{k \in \Lambda} w_k^{\lambda_{k,\Lambda}}.$$

5 Proposed Voting Protocol

The proposed voting protocol is a good combination of a typical mixnet voting protocol and the randomization technique introduced above. It consists of the following 5 stages.

Stage 1. System setup

Administrator A prepares system parameters of the threshold ElGamal encryption scheme. n talliers T_k jointly execute the key generation protocol of the (t,n)-threshold verifiable secret sharing scheme and publish the public key. Let $h = g^s$ be the public key and s be the private key shared by n talliers. A publishes the list of candidates and other required information on voting.

Stage 2. Registration

Voter V_i identifies and registers himself to A. A checks V_i's eligibility and issues TRR_i which is securely equipped with its own signing key and voter's public key. After the registration deadline has passed, A publishes the list of qualified voters with the certificates of voters and TRRs.

Stage 3. Voting

In the voting stage the voter V_i and his TRR_i compute the final encrypted ballot through the following interactive protocol.

– V_i prepares a ballot message m_i, chooses a random number $\alpha \in_R Z_q^*$, and computes a first ballot as $(x,y) = (g^\alpha, h^\alpha m_i)$. He sends (x,y) to TRR_i.
– TRR_i randomizes (x,y) to generate a final ballot $(x_f, y_f) = (xg^\beta, yh^\beta)$, where β is TRR's secure randomness, and signs it $Sig_{TRR_i}(x_f, y_f)$. It also computes a DVRP as described in Section 4. It computes $(a,b) = (g^k, h^k)$ and $d = g^r h_V^t$, where $k, r, t \in_R Z_q$, and computes $c = H(a,b,d,x_f,y_f)$ and $u = k - \beta(c+r)$. Then (c,r,t,u) is a DVRP for (x_f,y_f). TRR_i sends $Sig_{TRR_i}(x_f,y_f)$ and (c,r,t,u) to the voter.
– V_i verifies the validity of DVRP (c,r,t,u) by
$$c \overset{?}{=} H(g^u (x_f/x)^{c+r}, h^u (y_f/y)^{c+r}, g^r h_V^t, x_f, y_f).$$

If V_i is convinced that the final ballot is constructed correctly, V_i double signs the final ballot
$$Sig_{V_i}(Sig_{TRR_i}(x_f, y_f))$$
and posts it on the bulletin board.

Stage 4. Mixing

Before the mixing stage, A verifies the double signatures of voters and their TRRs from the posted ballots, and publishes valid ballots on the bulletin board. Then m mixers M_j shuffle the ballots sequentially and post the shuffled ballots on the bulletin board. In this stage previously proposed verifiable mixnet protocols such as [Abe99], [FS01], [Nef01], and [Gro03] can be used.

Stage 5. Tallying

For the shuffled ballots n talliers jointly decrypt each ballot using the (t, n)-threshold ElGamal decryption protocol to recover the original ballot messages m_i. Finally, A publishes the tally result.

6 Analysis

The proposed voting protocol satisfies all the security requirements proposed in Section 1.

- **Privacy**: The voter-vote relationship is hidden by the mixing service, so the privacy of voter depends on the security of mixnet. If talliers try to open a ballot before mixing, more than t talliers should cooperate. Assuming the honesty of at least $n - t + 1$ talliers, the secrecy of the vote is preserved.
- **Prevention of double voting**: Since the list of eligible voters are published and voters participate in voting in an authenticated manner (using their signature and TRR's signature), double voting is prevented.
- **Universal verifiability**: Since all the messages in each stage (voting, mixing, and tally) are published in the bulletin board and the correctness of processes is publicly verifiable, any observer can verify the validity of the vote result.
- **Fairness**: Assuming the honesty of at least $n - t + 1$ talliers that they will not cooperate to open the ballot before mixing, no partial tally is revealed, and fairness of voting is guaranteed.
- **Robustness**: Partial failure of some voters can be detected and it does not affect the whole voting protocol. Mixing and tally stages are robust against partial failure of the servers.
- **Receipt-freeness**: No voter can construct a receipt from the messages that he had sent or received, since he had lost his randomness. Although he is convinced that the vote message is preserved in the final ballot by DVRP, he cannot transfer the proof to others. Assuming that a buyer cannot observe the very moment of voter's voting and the communication channel between a voter and his TRR is internal, a voter cannot be coerced into casting a particular vote.

In some papers [Gol02, AI03], preventing ballot copying is discussed. In our construction, a voter can try to copy another voter's ballot because of the malleability of ElGamal encryption. If we want to prevent ballot copying, we have to require the voter to prove his knowledge of randomness. In our construction, voter cannot prove anything since he had lost his randomness. However, note that voter also cannot prove that he had copied a specific ballot (cannot construct a receipt).

The proposed method provides most mixnet-based voting protocols with receipt-freeness in a very efficient manner. All the computation required for TRR is 2 offline exponentiations for re-encryption, 4 offline exponentiations for DVRP creation, and 1 signing. Since all the computation in TRR is offline, it can be pre-computed and is suitable to implement in a hardware device which has limited computational power. On the other hand, the computation required for the voter is 6 online exponentiation for DVRP verification and 1 signature verification. Compared with the homomorphic encryption based receipt-free voting schemes, mixnet-based receipt-free voting is more efficient for voters since complex proof of validity of ballot is not needed.

Since tamper-resistant hardware device is the ultimate place to store user's secret information such as digital signing key, we expect that it will be available to many users in the near future and the proposed voting scheme will be quite reasonable.

7 Conclusion

In this paper we proposed a simple and efficient method to incorporate receipt-freeness in mixnet-based electronic voting schemes by using the well known re-encryption technique and DVRP. In our scheme a voter has to prepare his encrypted ballot through a randomization service provided by TRR in such a way that he finally loses his knowledge on the randomness. This method can be used in most mixnet-based electronic voting schemes to provide receipt-freeness in a very efficient manner.

However, we found that the efficient mixnet of Golle *et. al.* [Gol02] cannot be used in this construction. In their scheme double encryption is used to keep the privacy of vote in a bad case and to support backup mixing. But the inner encryption, which is known only to the voter, can work as a receipt. Applying our tool of receipt-freeness for those efficient mixnets is planned as future work.

Acknowledgements

We acknowledge the support of the Australian government through ARC Linkage-International fellowship scheme 2003, Grant No: LX0346868.

References

[Abe98] M. Abe, "Universally verifiable mix-net with verification work independent of the number of mix-servers", *Advances in Cryptology – Eurocrypt'98*, LNCS 1403, Springer-Verlag, pp. 437–447, 1998. 247

[Abe99] M. Abe, "Mix-networks in permutation networks", *Asiacrypt 1999*, LNCS 1716, Springer-Verlag, pp. 258–273, 1999. 247, 249, 255

[AI03] M. Abe, and H. Imai, "Flaws in some robust optimistic mixnets", *ACISP 2003*, LNCS 2727, Springer-Verlag, pp. 39–50, 2003. 249, 255

[Ben87] J. Benaloh, "Verifiable secret-ballot elections", PhD thesis, Yale University, Department of Computer Science, New Haven, CT, September 1987. 247

[Bau01] O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard and J. Stern, "Practical multi-candidate election system", *Proc. of the 20th ACM Symposium on Principles of Distributed Computing*, N. Shavit Ed., pp. 274–283, ACM Press, 2001. 247

[BT94] J. Benaloh and D. Tuinstra, "Receipt-free secret-ballot elections", *Proc. of 26th Symp. on Theory of Computing (STOC'94)*, pp. 544–553, New York, 1994. 247, 248

[CFSY96] R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung, "Multi-authority secret ballot elections with linear work", *Advances in Cryptology – Eurocrypt'96*, LNCS 1070, Springer-Verlag, pp. 72–83, 1996. 247

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure an optimally efficient multi-authority election schemes", *Advances in Cryptology – Eurocrypt'97*, LNCS 1233, Springer-Verlag, pp. 103–118, 1997. 247, 248, 253

[Cha88] D. Chaum, "Elections with unconditionally- secret ballots and disruption equivalent to breaking RSA", *Advances in Cryptology – Eurocrypt'88*, LNCS 330, Springer-Verlag, pp. 177–182, 1988. 247, 249

[ElG85] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Trans. on IT*, Vol.31, No.4, pp. 467–472, 1985. 252

[ES] "Electoral systems", Administration and Cost of Elections Project, available at http://www.aceproject.org/main/english/es/ 246

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale election", *Advances in Cryptology – Auscrypt'92*, LNCS 718, Springer-Verlag, pp. 244–260, 1992. 247

[FS01] J. Furukawa and K. Sako, "An efficient scheme for proving a shuffle", *Crypto 2001*, LNCS 2139, Springer-Verlag, pp. 368–387, 2001. 247, 249, 255

[Gol02] P. Golle, S. Zhong, D. Boneh, M. Jakobsson, and A. Juels, "Optimistic mixing for exit-polls", *Asiacrypt 2002*, LNCS 2501, Springer-Verlag, pp. 451–465, 2002. 247, 249, 255, 256

[Gro03] J. Groth, "A Verifiable Secret Shuffle of Homomorphic Encryptions", Appears in Public Key Cryptography - PKC 2003, LNCS 2567, Springer-Verlag, pp. 145–160, 2003. 247, 249, 255

[Hirt01] M. Hirt, "Multi-party computation: Efficient protocols, general adversaries, and voting", Ph.D. Thesis, ETH Zurich, Reprint as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre Verlag, Konstanz, 2001. 247, 248

[HS00] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption", *Advances in Cryptology - Eurocrypt2000*, LNCS 1807, Springer-Verlag, pp. 539–556, 2000. 247, 248

[Jak98a] M. Jakobsson, "A practical mix", *Advances in Cryptology – Eurocrypt'98*, LNCS 1403, Springer-Verlag, pp. 449–461, 1998. 247, 249

[Jak98b] M. Jakobsson, "Flash mixing", Proc. of the 18th ACM Symposium on PODC'98, pp. 83–89, 1998. 247, 249

[JSI96] M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications", *Advances in Cryptology – Eurocrypt'96*, LNCS 1070, Springer-Verlag, pp. 143–154, 1996. 252

[Kim01] K. Kim, J. Kim, B. Lee, and G. Ahn, "Experimental design of worldwide Internet voting system using PKI", *SSGRR2001*, L'Aquila, Italy, Aug. 6-10, 2001. 247

[LK00] B. Lee, and K. Kim, "Receipt-free electronic voting through collaboration of voter and honest verifier", *Proceeding of JW-ISC2000*, pp. 101–108, Jan. 25-26, 2000, Okinawa, Japan. 247, 248

[LK02] B. Lee, and K. Kim, "Receipt-free electronic voting scheme with a tamper-resistant randomizer", *ICISC 2002*, LNCS 2587, Springer-Verlag, pp. 389–406, 2002. 247, 248, 250

[MBC01] E. Magkos, M. Burmester, V. Chrissikopoulos, "Receipt-freeness in large-scale elections without untappable channels", *1st IFIP Conference on E-Commerce / E-business / E-Government*, Zurich, Octomber 2001, Kluwer Academics Publishers, pp. 683–693, 2001. 247, 248, 250

[Nef01] A. Neff, "A verifiable secret shuffle and its application to E-voting", *ACM CCS 2001*, ACM Press, pp. 116–125, 2001. 247, 249, 255

[Oka97] T. Okamoto, "Receipt-free electronic voting schemes for large scale elections", *Proc. of Workshop on Security Protocols'97*, LNCS 1361, Springer-Verlag, pp. 25–35, 1997. 248

[Ohk99] M. Ohkubo, F. Miura, M. Abe, A. Fujioka and T. Okamoto, "An improvement on a practical secret voting scheme", *Information Security'99*, LNCS 1729, Springer-Verlag, pp. 225–234, 1999. 247

[Pai99] P. Paillier, "Public-key cryptosystems based on discrete logarithms residues", *Advances in Cryptology - Eurocrypt '99*, LNCS 1592, Springer-Verlag, pp. 223–238, 1999.

[Ped91] T. Pedersen, "A threshold cryptosystem without a trusted party", *Advances in Cryptology - Eurocrypt '91*, LNCS 547, Springer-Verlag, pp. 522–526, 1991. 253

[PIK93] C. Park, K. Itoh, and K. Kurosawa, "Efficient anonymous channel and all/nothing election scheme", *Advances in Cryptology – Eurocrypt'93*, LNCS 765, Springer-Verlag, pp. 248–259, 1994. 247

[SK94] K. Sako and J. Kilian, "Secure voting using partial compatible homomorphisms", *Advances in Cryptology – Crypto'94*, LNCS 839, Springer-Verlag, pp. 411–424, 1994. 247

[SK95] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth", *Advances in Cryptology – Eurocrypt'95*, LNCS 921, Springer-Verlag, pp. 393–403, 1995. 247, 248

[Wik02] D. Wikstrom, "How to break, fix, and optimize optimistic mix for exit-polls", SICS Technical Report, T2002:24, available at http://www.sics.se/libindex.html, 2002. 249

Receipt-Free Electronic Auction Schemes Using Homomorphic Encryption

Xiaofeng Chen[1], Byoungcheon Lee[2], and Kwangjo Kim[1]

[1] International Research center for Information Security (IRIS)
Information and Communications University(ICU),
58-4 Hwaam-dong Yusong-ku, Taejon, 305-732 Korea
{crazymount,kkj}@icu.ac.kr
[2] Joongbu University,
101 Daehak-Ro, Chuboo-Meon, Kumsan-Gun, Chungnam, 312-702, Korea
sultan@joongbu.ac.kr

Abstract. Bid-rigging is a dangerous attack in electronic auction. Abe and Suzuki firstly introduced the idea of receipt-free to prevent this attack. In this paper we point out that Abe and Suzuki's scheme only provides receipt-freeness for losing bidders. We argue that it is more important to provide receipt-freeness for winners and propose a new receipt-free sealed bid auction scheme using the homomorphic encryption technique. In contrast to Abe and Suzuki's scheme, our scheme satisfies privacy, correctness, public verifiability and receipt-freeness for all bidders. Also, our scheme is not based on *threshold trust model* but *three-party trust model*, so it is more suitable for real-life auction. Furthermore, we extend our scheme to $M + 1$-st price receipt-free auction.

Keywords. Bid-rigging, Receipt-free, Homomorphic encryption, Auction.

1 Introduction

Auction has become a major phenomenon of electronic commence in the recent years. The most common auctions are English auction, Dutch auction and Sealed bid auction. In the former two auctions, the communication cost is high and the bidder information will be revealed after the auction is finished. Though the sealed bid auction can be finished in one round communication, it does not support the optional distribution of the goods [11]. Nobel Prize winner, economist Vickrey presented a second-price auction: the bidder with the highest price wins, but he only pays the second-price [25]. Vickrey auction is celebrated in economics for having the property of *incentive compatibility*, *i.e.*, the dominant strategy for each bidder is always to bid his true value. However, it is rarely used in practice for some crucial weakness [17]. $M + 1$-st price auction is a type of sealed-bid auction for selling M units of a single kind of goods. In this auction, M highest bidders win and only pay $M + 1$-st winning price. Vickrey auction can be regarded as a special case of $M + 1$-st price auction for $M = 1$. Wurman [26] proved that $M + 1$-st price auction also satisfies the property of *incentive compatibility*.

Sakurai *et al.* [22] firstly pointed out there exists the problem of bid-rigging in the electronic auction, *i.e.*, coercer/buyer orders other bidders to bid the price he specifies to control the winning price. For instance, the coercer/buyer orders other bidders (victims) to bid very low prices, then can win the auction at an unreasonably low price. Hence, if bid-rigging occurs, the auction fails to establish the appropriate price, so it is important to prevent bid-rigging. Sakurai *et al.* presented an anonymous auction protocol based on a new convertible group signature scheme to hide the identity of the winner. However, it does not solve the bid-rigging essentially. Recently, Abe and Suzuki [2] introduced the idea of receipt-free to prevent bid-rigging in the auction protocol.[1] They also proposed a receipt-free sealed-bid auction scheme under the assumption of bidding booth and a one-way untappable channel. However, the scheme is based on *threshold trust model*, *i.e.*, a major fraction of auctions are assumed to be honest, which is not suitable for many real life electronic auctions [5, 17, 18]. Also, it is not suit for $M+1$-st price auction.

Moreover, in Abe and Suzuki's scheme, all auctioneers together recover the secret seeds of each bidder to determine the winning price and the winners, that is, the secret identity number j of the winners is revealed to all auctioneers. Therefore a dishonest auctioneer may tell this information to the coercer/buyer. The coercer/buyer can know all victims' secret identity number beforehand. If the winner is one of the victims he ordered, the coercer/buyer will punish the winner. Therefore, the scheme only provides the receipt-freeness for losing bidders. We argue that it is more important to provide receipt-freeness for winners in electronic auctions. The aim of a coercer/buyer is to win the auction and the losers do not affect the result of the auction. So, if the victim is not the winner, the coercer/buyer never care whether the victim cheats him or not. However, if a bidder violates the rule of coercer/buyer and wins the auction, the coercer/buyer will be sure to punish the bidder for he cannot get the auction item.

In this paper, we propose a new receipt-free sealed bid auction scheme based on homomorphic encryption technique. In contrast to Abe and Suzuki's scheme, our scheme is not based on *threshold trust model* but *three-party trust model* [18]. Our scheme satisfies the properties of secrecy, correctness, public verifiability and receipt-freeness for all bidders. Furthermore, we present receipt-free $M+1$-st price auction.

The rest of the paper is organized as follows: The next section gives some properties and security requirements of an electronic auction. Section 3 proposes the receipt-free sealed bid auction scheme. Section 4 presents the receipt-free $M+1$-st auction scheme. In Section 5, the security and efficiency analysis about our proposed auction scheme are presented. Section 6 concludes this paper.

[1] The concept of receipt-free is firstly introduced by Benaloh and Tuinstra [4] to solve the misbehaver of "vote buying" or "coercion" in the electronic voting. There are plenty of researches on receipt-freeness in the electronic voting [12, 19, 21].

1.1 Related Works

There are plenty of works on the sealed bid (highest price) auction and Vickrey auction. Franklin and Reiter [10] presented a protocol for implementing a sealed-bid auction. Their solution is focused on using a cryptographic technique to provide protections to monetary bids, such as digital money. In their protocol, auctioneers use atomic multicast to communicate with each other, which is a bottleneck in a large system. Also the protocol has the disadvantage of revealing the bids after the auction is finished. Kikuchi *et al.* [11] proposed a protocol for a sealed-bid auction based on the Shamir's secret sharing scheme. The scheme enjoys the privacy of the bids, but it does not guarantee that the winning bidder will pay for the bid item. These schemes are not well suited for handing tie bids, *i.e.*, when two or more bidders happen to submit the same winning bid [20]. If tie-bidding occurs, the schemes will not specify who the winners are, or even how many winners there are. Also, both of the schemes distribute the trust onto multi auctioneers. Recently, Cachin [6] proposed an auction scheme involving two auction servers, but requiring bidders to contact just a single server. Lipmaa *et al.* [17] proposed a secure Vickrey auction without threshold trust.

The $M+1$-st price auction is a type of sealed-bid auction for selling M units of a single kind of item. Due to its attractive property of incentive compatibility, there are many works on $M+1$-st price auction [2, 5, 14]. Kikuchi [14] proposed an $M+1$-st price auction using homomorphic encryption. Brandt [5] proposed an $M+1$-st price auction where bidders compute the result by themselves. However, there is no receipt-free $M+1$-st price auction to the best of our knowledge.

1.2 Trust Model

There are numerous researches on electronic auction in recent years. Based on the trust model, the researches can be classified into the following cases:

Threshold Trust Model. There are m auctioneers in threshold trust model, out of which a fraction (e.g. more than $m/3$ or $m/2$) are assumed to be trustworthy. The auctioneers jointly compute the winning price by using inefficient techniques of secure multiparty function evaluation [18]. Some researchers claimed that the threshold trust model is not suitable for real-life electronic auctions [5, 17, 18, 23].

Three-Party Trust Model. A new third-party is introduced in this trust model. The third-party is not a fully trusted party but assumed not to collude with the auctioneer [17, 18] or other parties [3, 6]. Also, the third-party is not required to interact with bidders and just generates the program for computing the result of the auction.

No Auctioneers Trust Model. Brandt [5] introduced the concept of bidder-resolved auctions, which distribute the trust onto all of the bidders. Unless all involved parties collude, no information of the bids will be revealed. It is a reasonable assumption that all bidders will never share their information simultaneously due to competition among them. A drawback of this model is low efficiency.

2 Properties and Security Requirements of Auction

In this section, we briefly describe the properties and security requirements of the electronic auction.

2.1 Properties

In this paper, we focus on sealed-bid auctions: bids are keep secret during the bidding phase. In the bidding phase, the bidders sends their sealed bidding price. In the opening phase, the auctioneer (or with a third party) determines the winning price according to a predetermined auction rule.

– **Bidding:** The auctioneer advertises the auction and calls the bidders to bid for the auction item. Each bidder decides his bidding price and sends the sealed price (better in an "encrypted" manner) to the auctioneer. The auctioneer cannot recover the information about the bids.
– **Opening:** After all bidders have sent their bidding price, the auctioneer determines the winning price and finds the bidder who bids the highest price. The information of the loser's identity and bidding price should not be revealed even after the auction. Furthermore, it is crucial to protect the identity of the winners in receipt-free auction. Otherwise, a coercer/buyer will punish the winners who do not bid the specified price.
– **Trading:** The winner buys the auction item with a certain price according to the auction rule. If the winner wants to repudiate his price, the auctioneer (or collaborating with other entities) can identity the winner.

2.2 Security Requirements

– **Privacy**: No information of the bids and the corresponding bidder's identity is revealed during and after the auction. The only information to be revealed is the winning price.
– **Correctness**: The winner and the winning price are determined correctly by a certain auction rule.
– **Public verifiability**: Anyone can verify the correctness of the auction.
– **Non-repudiation**: No bidder can repudiate his bid. The winner must buy the auction item with a certain price according the auction rule.
– **Efficiency**: The communication and computation in the auction should be reasonable for implementation.
– **Receipt-freeness**: Anyone, even if the bidder himself, must not be able to prove any information about the bidding price to any party.

3 Receipt-Free Sealed Bid Auctions

In this section, we present receipt-free sealed bid auctions. Our scheme is not based on threshold trust model but three-party trust model, where the entity called *Auction Issuer* is not fully trusted but assume not collude with Auctioneer.

3.1 Physical Assumptions

Bulletin Board: In Abe and Suzuki's scheme, a physical assumption called bidding booth is used: no one can control or watch the bidder in the bidding booth. In our scheme, we will use a weaker assumption called bulletin board: any one can read the information in the bulletin board, but no one can delete the information. Also, only active bidders can append information in the designated fields. We suppose the bidders who register with auction service are assigned a secret identity number and the corresponding fields in the bulletin board. As Stubblebine *et al.* [23] discussed, registration may incorporate an identity escrow mechanism [15] so that the identity of a winner might only be revealed if he repudiates his price.

Untappable Channel: This is a one-way communication channel and the message through this channel remains secret for any third party. In our scheme, we assume that two untappable channels from auctioneer issuer and auctioneer to seller are separately available. Also, there exist untappable channels between each bidder and seller. Therefore, the coercer/buyer can not know the communication between the bidders and the seller, *i.e.*, he can not control the victims during the bidding.

3.2 System Parameters

There are m bidders $\{B_i | i = 1, 2, \cdots, m\}$, a seller S, an auctioneer A and an auctioneer issuer AI. Consider the subgroup G_q of order q of Z_p^*, where p and $q|p-1$ are large primes and g is a generator of G_q. Let G_1 and G_2 be independently selected generators of G_q which mean "I bid" and "I do not bid", respectively.

- A: chooses his secret key x_1 and publishes his public key $h_1 = g^{x_1}$.
- AI: chooses his secret key x_2 and publishes his public key $h_2 = g^{x_2}$.
- S: publishes a price list $P = \{j | j = 1, 2, \cdots, n\}$.
- B_i: chooses his secret key x_{B_i} and publishes his public key $h_{B_i} = g^{x_{B_i}}$. He decides his bidding price $p_i \in P$ and computes the encrypted bidding vector

$$C_{i,j} = (x_{i,j}, y_{i,j}) = \begin{cases} (g^{a_{ij}}, (h_1 h_2)^{a_{i,j}} G_1), & \text{if } j = p_i \\ (g^{a_{ij}}, (h_1 h_2)^{a_{i,j}} G_2), & \text{if } j \neq p_i \end{cases}$$

  where $a_{i,j} \in_R Z_q$, $j = 1, 2, \cdots, n$.

3.3 High-Level Description of the Scheme

For $j = 1, 2, \cdots, n$, each bidder B_i firstly sends the bidding vector $C_{i,j}$ to the seller S. Then the seller S generates the receipt-free biding vector $C^*_{i,j} = (x^*_{i,j}, y^*_{i,j}) = (x_{i,j}u_j, y_{i,j}v_j)$, where $u_j = g^{\beta_j}$ and $v_j = (h_1h_2)^{\beta_j}$. Meanwhile, the seller S proves that u_j and v_j have the common exponent β_j without exposing the value of β_j with the designated-verifier re-encryption knowledge proof [16]. If the proof is valid, the bidder B_i and the seller S jointly generate a proof of the validity of the bidding vector $C^*_{i,j}$ (see Appendix B and C). The bidder then posts the bidding vector and the proof to the designated fields in the bulletin board.

In the opening phase, the auctioneer and the Auctioneer Issuer together determine the winning (highest) price with $\prod_{i=1}^{m} C^*_{i,j}$ for $j = n, n-1, \cdots, 1$. Unless the auctioneer and the Auctioneer Issuer collude, the bidder's privacy will not be revealed.

3.4 Proposed Receipt-Free Sealed Bid Auction Scheme

In our scheme, we do not rely on any trusted party. Assume that auctioneer will never work in tandem with Auctioneer Issuer. Also, we argue that the seller S will never collude with a coercer/buyer because the seller is *benefit-collision* with the coercer/buyer. If the bid-rigging occurs, the auction fails to establish the appropriate price and the coercer/buyer may win the auction at an unreasonable low price. So, the bid-rigging is benefit to the coercer/buyer while the seller suffers a great loss.

Bidding: Each bidder B_i generates the receipt-free biding vector $C^*_{i,j}$ with the help of the seller S, where $j = 1, 2, \cdots, n$. Also, they jointly generate the proof of the validity of the bidding vector.

– B_i sends $C_{i,j} = (x_{i,j}\,, y_{i,j})$ to S, where $j = 1, 2, \cdots, n$.
– For $j = 1$ to n, S chooses β_j and then computes $u_j = g^{\beta_j}$ and $v_j = (h_1h_2)^{\beta_j}$. Let the receipt-free biding vector is $C^*_{i,j} = (x^*_{i,j}, y^*_{i,j}) = (x_{i,j}u_j, y_{i,j}v_j)$.
– For $j = 1$ to n, S chooses $\gamma_j, \zeta_j, \delta_j \in_R Z_q$ and computes

$$(a_j,\, b_j) = (g^{\gamma_j},\, (h_1h_2)^{\gamma_j}),\ D_j = g^{\zeta_j} h_{B_i}^{\delta_j}$$

– S computes $H_j = H(a_j, b_j, D_j, x^*_{i,j}, y^*_{i,j})$, $U_j = \gamma_j - \beta_j(H_j + \zeta_j)$. He then sends $(H_j,\, \zeta_j,\, \delta_j,\, U_j)$ and $C^*_{i,j}$ to B_i.
– B_i verifies the equation

$$H_j = H(g^{U_j}(x^*_{i,j}/x_{i,j})^{H_j+\zeta_j}, (h_1h_2)^{U_j}(y^*_{i,j}/y_{i,j})^{H_j+\zeta_j}, g^{\zeta_j}h_{B_i}^{\delta_j}, x^*_{i,j}, y^*_{i,j})$$

holds.
– For $j = 1$ to n, B_i chooses $w_j, d_j, r_j \in_R Z_q$.

– If $j = p_i$, B_i computes

$$a_{1,j} = g^{w_j},\ b_{1,j} = (h_1h_2)^{w_j},\ a_{2,j} = g^{r_j}x_{i,j}^{d_j},\ b_{2,j} = (h_1h_2)^{r_j}(y_{i,j}/G_2)^{d_j};$$

else, B_i computes

$$a_{1,j} = g^{r_j}x_{i,j}^{d_j},\ b_{1,j} = (h_1h_2)^{r_j}(y_{i,j}/G_1)^{d_j},\ a_{2,j} = g^{w_j},\ b_{2,j} = (h_1h_2)^{w_j};$$

– B_i sends $(a_{1,j}, b_{1,j})$ and $(a_{2,j}, b_{2,j})$ to S, where $j = 1, 2, \cdots, n$.
– For $j = 1$ to n, S chooses $r_{1,j}, r_{2,j}, d_{1,j} \in Z_q$
– S computes $a'_{1,j} = a_{1,j}g^{r_{1,j}}x_{i,j}^{d_{1,j}}$, $b'_{1,j} = b_{1,j}(h_1h_2)^{r_{1,j}}(y_{i,j}/G_1)^{d_{1,j}}$, $a'_{2,j} = a_{2,j}g^{r_{2,j}}x_{i,j}^{-d_{1,j}}$, $b'_{2,j} = b_{2,j}(h_1h_2)^{r_{2,j}}(y_{i,j}/G_2)^{-d_{1,j}}$.
– S sends $P_1 = (a'_{1,j}, b'_{1,j}, a'_{2,j}, b'_{2,j})$ to B_i
– B_i computes $c_j = H(a'_{1,j}, b'_{1,j}, a'_{2,j}, b'_{2,j})$, $e_j = c_j - d_j$, $f_j = w_j - a_{ij}e_j$, where $j = 1, 2, \cdots, n$.
– Let $X = e_{p_i}$, $e_{p_i} = d_{p_i}$, $d_{p_i} = X$; $Y = f_{p_i}, f_{p_i} = r_{p_i}, r_{p_i} = Y$;
– For $j = 1$ to n, B_i sends (d_j, r_j), (e_j, f_j) to S.
– For $j = 1$ to n, S computes $d'_{1,j} = d_j + d_{1,j}$, $d'_{2,j} = e_j - d_{1,j}$, $r'_{1,j} = r_j + r_{1,j} - d'_{1,j}\beta_j$, $r'_{2,j} = f_j + r_{2,j} - d'_{2,j}\beta_j$ and sends $P_2 = (d'_{1,j}, d'_{2,j}, r'_{1,j}, r'_{2,j})$ to B_i.
– B_i computes $c_j = d'_{1,j} + d'_{2,j}$ and verifies
$$c_j = H(g^{r'_{1,j}}(x^*_{i,j})^{d'_{1,j}}, (h_1h_2)^{r'_{1,j}}(y^*_{i,j}/G_1)^{d'_{1,j}},$$
$$g^{r'_{2,j}}(x^*_{i,j})^{d'_{2,j}}, (h_1h_2)^{r'_{2,j}}(y^*_{i,j}/G_2)^{d'_{2,j}})$$
– B_i sends $C^*_{i,j}$, P_1 and P_2 to the corresponding fields of the bulletin board.

Opening: AI and A compute the auction result.

– Let $j = n$, AI and A compute separately the final price vector

$$(X_j, Y_j) = (\prod_{i=1}^{m} x^*_{i,j}, \prod_{i=1}^{m} y^*_{i,j})$$

They then separately publish $X_j^{x_1}$, $X_j^{x_2}$ and provide a non-interactive zero-knowledge proof of common exponent with their public key h_1, h_2. Let

$$R_j = Y_j/X_j^{x_1+x_2} = G_1^{l_j}G_2^{m-l_j}$$

where $0 \le l_j \le m$.[3] If $l_j = 0$, $j = j - 1$; else terminated.
– AI and A determine the first j which satisfies $l_j \ne 0$, and the winning price is the certain j, denote P_w.[4]
– AI and A publish the winning price P_w.

[3] From the result of [8], we know the complexity of computing l_j is $m^{1/2}$. Therefore, m can not be very large, *i.e.*, our scheme is unfit for very large scale auction

[4] If $l_j > 1$, the case of tie-bids occurred for there are l_j winner candidates. However, compare with the previous schemes, we know the numbers of winner candidates. Then these candidates perform the next round of auction.

On the other hand, if n is chosen reasonable large, the bidders can submit the price as his will so that the tie-bids can be avoided.

Trading: The winner proves that his bidding price is P_w and buys the auction item. The winner can not repudiate his price, because S can identity the winner with the help of AI and A.

- The winner B_i sends C_{i,P_w} to the seller.
- Since S knows all $C_{i,j}$ for $j = 1, 2, \cdots, n$, and $i = 1, 2, \cdots, m$, he can check the validity of C_{i,P_w} easily. If C_{i,P_w} is valid, go to the next step; else, terminated.
- The winner provides a knowledge of common exponent to x_{i,P_w} and $y_{i,P_w}/G_1$.
- If the winner wants to cancel the trading, AI and A compute $(x^*_{i,P_w})^{x_1}$ and $(x^*_{i,P_w})^{x_2}$, respectively, where $i = 1, 2, \cdots, m$. So, S can identity the winner i which satisfies $G_1 = y^*_{i,P_w}/(x^*_{i,P_w})^{x_1+x_2}$. S sends the number of the corresponding fields in the bulletin board to the auction service and the service publics the identity of the winner. The winner must be answered for his dishonest deeds, for example, the seller announces his name in the black lists, or the down payment will be expropriated.

4 Receipt-Free $M + 1$-st Price Auctions

In this section we will extend our scheme for $M + 1$-st price auction. The above scheme cannot be extended to $M + 1$-st price directly because the winner must prove his bidding price to S in the trading, *i.e.*, S will know all the winners' price.

Abe and Suzuki [1] proposed an $M + 1$-st price auction using homomorphic encryption, which enjoys privacy, public verifiability. We will extend this for receipt-free $M + 1$-st auction. Firstly, we introduce some definitions as [1].

Definition 1. *Let $E_a(M)$ denotes $(g^a, h^a M)$. Define*

$$E_a(M)E_b(N) = (g^{a+b}, h^{a+b}MN)$$

and

$$(E_a(M))^{-1} = (g^{-a}, h^{-a}M^{-1})$$

Definition 2. *Given a vector*

$$A(j) = (E_{a_1}(M), \cdots, E_{a_j}(M), E_{a_{j+1}}(1), \cdots, E_{a_n}(1))$$

the "differential" vector $\Delta A(j)$ of A_j is defined

$$\Delta A(j) = (E_{b_1}(1), \cdots, E_{b_{j-1}}(1), E_{b_j}(M), E_{b_{j+1}}(1), \cdots, E_{b_n}(1))$$

and satisfies

$$A(j)_n = \Delta A(j)_n, A(j)_{n-1} = \Delta A(j)_{n-1}A(j)_n, \cdots, A(j)_1 = \Delta A(j)_1 A(j)_2$$

where $A(j)_i$ and $\Delta A(j)_i$ denote the i-th components of vectors $A(j)$ and $\Delta A(j)$, respectively. We call $A(j)$ the "integral" vector of $\Delta A(j)$. Therefore, given the "differential" vector of a vector, the vector can be recovered efficiently, vice versa.

In general, we have

Definition 3. *Given a vector*

$$A(j) = (E_{a_1}(M), \cdots, E_{a_j}(M), E_{a_{j+1}}(N), \cdots, E_{a_n}(N))$$

the "differential" vector $\Delta A(j)$ *of* A_j *is defined*

$$\Delta A(j) = (E_{b_1}(N), \cdots, E_{b_{j-1}}(N), E_{b_j}(M), E_{b_{j+1}}(N), \cdots, E_{b_n}(N))$$

and satisfies

$$A(j)_n = \Delta A(j)_n, A(j)_{n-1} = \Delta A(j)_{n-1} A(j)_n (A(j)_n)^{-1} = \Delta A(j)_{n-1}$$

$$A(j)_{n-2} = \Delta A(j)_{n-2} A(j)_{n-1} (A(j)_n)^{-1}, \cdots, A(j)_1 = \Delta A(j)_1 A(j)_2 (A(j)_n)^{-1}$$

where $A(j)_i$ *and* $\Delta A(j)_i$ *denote the i-th components of vectors* $A(j)$ *and* $\Delta A(j)$*, respectively. Therefore, given the "differential" vector of a vector, the vector can be recovered efficiently, vice versa.*

In the $M+1$-st price auction, the bidder B_i proves not only that each component of the bidding vector suits the form of $E(G_1)$ or $E(G_2)$ but also that there is only one component is the form of $E(G_1)$. Otherwise, a malicious bidder will submit an invalid bidding vector to destroy the result of the auction.[5]

We will construct our receipt-free $M+1$-st price auction scheme based on Abe and Suzuki's scheme:

Bidding: Each bidder B_i generates the receipt-free biding vector with the help of the seller S. They then jointly generate the proof of validity of the bidding vector. This is same with the first price sealed bid auction but adding a knowledge proof of that there is only one component in the bidding vector is the form of $E(G_1)$: the bidder B_i proves S that $\prod_{j=1}^{n} x_{i,j}$ and $\prod_{j=1}^{n} y_{i,j}/(G_1 G_2^{n-1})$ have the same exponent.

Opening: AI and A compute the $M+1$-st price.

- AI and A compute the "integral" vector $C_{i,j}^{**} = (x_{i,j}^{**}, y_{i,j}^{**})$ of $C_{i,j}^{*}$.
- Let $j=1$, AI and A compute separately

$$(X_j, Y_j) = (\prod_{i=1}^{m} x_{i,j}^{**}, \prod_{i=1}^{m} y_{i,j}^{**})$$

Then they publish $X_j^{x_1}$ and $X_j^{x_2}$, respectively and provide a non-interactive knowledge proof of common exponent with their public key h_1 and h_2. Let

$$R_j = Y_j / X_j^{x_1+x_2} = G_1^{l_j} G_2^{m-l_j}$$

where $0 \leq l_j \leq m$. If $l_j \geq M+1$, $j = j+1$; else terminated.

[5] In the first price sealed bid auction, AI and A determine the first j which satisfies $l_j \neq 0$ for $j = n, n-1, \cdots, 1$, and the winning price is j. Then the protocol is terminated. Therefore, the bidder B_i only proves that each component of the bidding vector suits the form of $E(G_1)$ or $E(G_2)$ and the result of the auction will not be affected.

- AI and A determine the first j which satisfies $l_{j-1} \geq M+1$ and $l_j \leq M$, and the winning price is the certain j, denote P_w.
- AI and A publish the winning price P_w.

Trading: The winners prove that their bidding price is large than P_w and buys one of the auction items at the price of P_w. The winners can not repudiate his price, because S can identity them with the help of AI and A.

- The winner B_i sends $C_{i,j}$ to the seller.
- The seller firstly check the validity of $C_{i,j}$ and then compute the "integral" vector vector $C^{'}_{i,j}$ of $C_{i,j}$.
- The winner B_i provides a knowledge proof of common exponent to $x^{'}_{i,P_w}$ and $y^{'}_{i,P_w}/G_1$.
- If some winners want to cancel the trading, for $i = 1, 2, \cdots, m$, AI and A compute $(x^{**}_{i,P_w})^{x_1}$ and $(x^{**}_{i,P_w})^{x_2}$, respectively and then send them to S via the untappable channel.
- S can identity the winner i which satisfies $G_1 = y^{**}_{i,P_w}/(x^{**}_{i,P_w})^{x_1+x_2}$.

5 Analysis of the Proposed Scheme

5.1 Security

The proposed scheme satisfies the following properties:

Privacy. In the first price auction, only the highest price is computed. Unless the auctioneer A and auctioneer issuer AI collude, the information of all losers remains secret. In the $M+1$-st price auction, only the $M+1$-st price is revealed, all other bidding price is secret. Also, the winner remains anonymous unless he wants to repudiate his bidding.

Correctness. It is trivial. No malicious bidders can affect the result of the auction due to the interactive proof of knowledge, which ensures each bidder must generate a valid bidding vector.

Public verifiability. Any entity can obtain the information to verify the correctness of the auction from the designated fields in the bulletin board.

Non-repudiation. With the help of both AI and A, S can know the designated field of the winners in the bulletin board. So, the winner cannot repudiate his bidding.

Receipt-freeness. In our scheme, $D_j = g^{\zeta_j} h_{B_i}^{\delta_j}$ is a trapdoor commitment (or chameleon commitment). Since B_i knows his private key x_{B_i}, he can compute ζ'_j and δ'_j such that $\zeta'_j + x_{B_i}\delta'_j = \zeta_j + x_{B_i}\delta_j$, *i.e.*, he can freely open the commitment as he wants and generate the re-encryption proof for any bidding. For details, see appendix B. Also, we suppose that S will not collude with the coercer/buyer. The coercer/buyer can not know the secret β_j of S. Therefore, the designated-verifier re-encryption proof can not be used to construct a receipt.

	Pattern	*Round*	*Volume*
Bidding (bidding vector)	$B_i \leftrightarrow S$	2m	O(n)
Bidding (proof)	$B_i, S \rightarrow$ *Bulletin board*	m	O(n)
Opening	$AI, A \rightarrow$ *Bulletin board*	*at most* m	$O(1)$
Trading (normal case)	*Winner* $\rightarrow S$	1	$O(1)$
Trading (repudiation case)	$AI, A \rightarrow S$	*at most* m	$O(1)$

Table 1. The communication complexity of our scheme

	Computational Complexity
One Bidder	n *encryptions and proofs*
Seller	mn *verfications and proofs*
A and AI	*at most* 2mn *multiplications*, n *decryptions and verifications*

Table 2. The computation complexity of our scheme

5.2 Efficiency

In our schemes, the seller S is involved in the auction to help the bidders to construct the receipt-free bidding. In the previous schemes, this work should be done by the auctioneer. However, there is no special trusted parities in our scheme and the auctioneer may collude with the coercer/buyer. We argue that it is the seller mainly suffered from the bid-rigging so we only trust that the seller will never collude with the coercer/buyer. The auctioneer is responsible for advertising the auction, computing the result of the auction and identifying the winner who wants to repudiate his price with the help of the auctioneer issuer. The auctioneer will not be the bottleneck of the auction. Furthermore, the seller should pay less for the auction since he shares some work of the auctioneer.

In the following we analyze the computation and communication of our scheme (the first price auction). Let n and m represent the number of bidding prices and bidders, respectively. Note that the computation and communication as shown in the tables are for all bidders, not for each bidder.

Table 1 presents the communication patterns, the numbers of the rounds and volume per round in the proposed scheme.

Table 2 shows the computation complexity of our scheme. It is easy to see that the efficiency of the proposed scheme is comparable to that of Abe and Suzuki's scheme.

On the other hand, the complexity of each bidder is proportional to n and the complexity of the seller is proportional to mn. Therefore, as we have mentioned above, our scheme is unsuitable for large scale auction and the price range should be reasonable large.

6 Conclusion

Bid-rigging is a dangerous attack in electronic auction. Abe and Suzuki firstly introduced the idea of receipt-free auction to prevent this attack. We point out that their auction scheme only provides receipt-freeness for losing bidders because the secret identity number of the winner can be revealed to the coercer/buyer by the dishonest auctioneers. Also, their scheme is not suitable for $M+1$-st price auction. We claim that it is more important to provide receipt-freeness for winners in the electronic auction. In this paper we propose a new receipt-free sealed bid auction scheme based on three-party trust model by using homomorphic encryption technique, and we extend it suitable for $M+1$-st price auction. Our scheme provides privacy, correctness, public verifiability and receipt-freeness for all bidders.

Acknowledgement

The first author is grateful to K. Suzuki for his meaningful discussion and valuable comments to this paper.

References

[1] M. Abe and K. Suzuki, *M+1-st Price Auction using Homomorphic Encryption*, Proceedings of Public Key Cryptography 2002, LNCS 2274, pp.115-124, Springer-Verlag, 2002. 266

[2] M. Abe and K. Suzuki, *Receipt-Free Sealed-Bid Auction*, ISC 2002, LNCS 2433, pp.191 -199, Springer-Verlag, 2002. 260, 261

[3] O. Baudron and J. Stern, *Non-interactive Private Auctions*, Proceedings of Financial Cryptography 2001, LNCS 2339, pp.364-377, Springer-Verlag, 2002. 261

[4] J. Benaloh, D. Tuinstra, *Receipt-free secret-ballot elections*, In Proc. of 26th Symp. On Theory of Computing (STOC'94), pp.544-553, 1994. 260

[5] F. Brandt, *Secure and private auctions without auctioneers*, Technical Report FKI-245-02, Institut fur Informatik, Technische Universitat Munchen, 2002. 260, 261, 262

[6] C. Cachin, *Efficient private bidding and auctions with an obvilious third party*, Proceeding of the 6th ACM Conference on Computer and Communications Security, pp.120-127, 1999. 261

[7] R. Cramer, I. Damgkd and B. Schoenmakers, *Proofs of partial knowledge and simplified design of witness hiding protocols*, Advances in Cryptology-CRYPTO 1994, LNCS 839, pp.174-187, Springer-Verlag, 1994.

[8] R. Cramer, R. Gennaro and B. Schoenmakers, *A Secure and Optimally Efficient Multi- Authority Election Scheme*, Advances in Cryptology - EUROCRYPT 1997, LNCS 1233, pp.103-118, 1997. 265

[9] D. Chaum and T. P. Pedersen, *Wallet databases with observers*, Advances in Cryptology-CRYPTO 1992, LNCS 740, pp.89-105, Springer-Verlag, 1993. 271

[10] M. K. Franklin and M. K. Reiter, *The design and implementation of a secure auction server*, IEEE Trans. on Software Engineering, 22(5), pp.302–312, 1996. 261

[11] M. Harkavy, H. Kikuch and J. D. Tygar, *Electronic Auction with Private Bids*, Proceeding of the 3rd USENIX Workshop on Electronic Commence, 1998. 259, 261

[12] M. Hirt and K.Sako, *Efficient receipt-free voting based on homomorphic encryption*, Advances in Cryptology-EUROCRYPT 2000, LNCS 1807, pp.393-403, Springer-verlag, 2000. 260

[13] A. Juels and M. Szydlo, *A Two-Sever, Sealed-Bid Auction Protocol*, Proceedings of Financial Cryptography 2002, LNCS 2357, Springer-Verlag, 2002.

[14] H. Kikuchi, *(M+1)st-Price Auction Protocol*, Proceedings of Financial Cryptography 2001, LNCS 2339, pp.351-363, Springer-Verlag, 2002. 261

[15] M. Kilian and E. Pertrank, *Identity Escrow*, Advance in Cryptology-CRYPTO 1998, LNCS 1462, pp.169-185, Springer-Verlag, 1998. 263

[16] B. Lee and K. Kim, *Receipt-free electronic voting scheme with a tamper-resistant randomizer*, ICISC 2002, LNCS 2587, pp.389-406, Springer-Verlag, 2002. 264

[17] H. Lipmaa, N. Asokan and V. Niemi, *Secure Vickrey Auctions without Threshold Trust*, Proceedings of Financial Cryptography 2002, LNCS 2357, pp.87-101, Springer-Verlag, 2002. 259, 260, 261

[18] M. Naor, B. Prinkas and R. Summer, *Privacy Preserving Auctions and Mechanism Design*, Proceedings of ACM conference on E-commerce, pp.129-139, Springer-Verlag, 1999. 260, 261

[19] T. Okamoto, *Receipt-free electronic voting schemes for large scale elections*, In Proceeding of Workshop on Security Protocols'97, LNCS 1361, pp.25-35, Springer-Verlag, 1997. 260

[20] K. Sako, *An Auction Protocol Which Hides Bids of Losers*, Proceedings of Public Key Cryptography 2000, LNCS 1751, pp.422-433, Springer-Verlag, 2000. 261

[21] K. Sako and J. Kilian, *Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth*, Advance in Cryptology-EUROCRYPT'95, LNCS 921, pp.393-403, Springer-verlag, 1995. 260

[22] K. Sakurai and S. Mkiyazaki, *An Anonymous Electronic Bidding Protocol Based on a New Convertible Group Signature Scheme*, ACISP 2000, LNCS 1841, Springer-Verlag, pp.385-399, 2000 260

[23] S. G. Stubblebine and P. F. Syverson, *Fair On-line Auction without Special Trusted Parties*, Proceedings of Financial Cryptography 1999, LNCS 1648, pp.230-240, Springer-Verlag, 1999. 261, 263

[24] K. Suzuki, K. Kobayashi and H. Morita, *Efficient Sealed-bid Auction Using Hash Chain*, ICICS 2000, LNCS 2015, pp.183-191, Springer-Verlag, 2000.

[25] W. Vickrey, *Counterspeculation, Auctions, and Competitive Sealed Tenders*, Journal of Finance, pp.8-37, 1961. 259

[26] P.R. Wurman, W.E. Walsh and M.P. Wellman, *Flexible Double Auctions for Electronic Commerce: Theory and Implementation*, Decision Support Systems, 24, pp.17-27, 1998. 259

Appendix A: Proof of Knowledge of Common Exponent

A prover with possession a secret number $\beta \in Z_q$ wants to show that $\log_g u = \log_h v$ while without exposing β, where $u = g^\beta$, $v = h^\beta$. Chaum and Pedersen [9] firstly proposed an interactive protocol to solve this problem.

Let $c = H(a, b, u, v)$, the above protocol could be easily converted into a non-interactive proof of knowledge, where $H()$ is one way hash function.

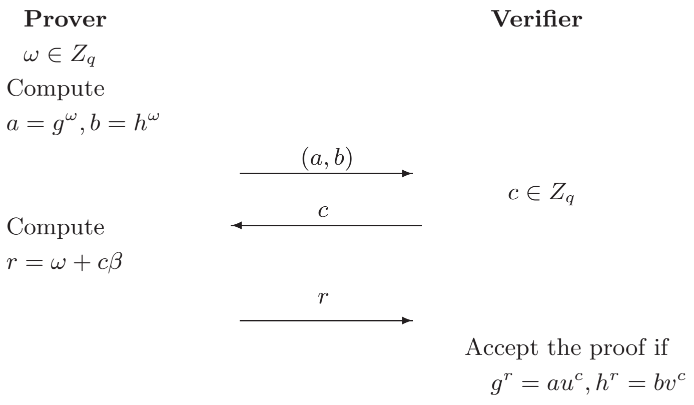


Fig. 1. Proof of knowledge of common exponent

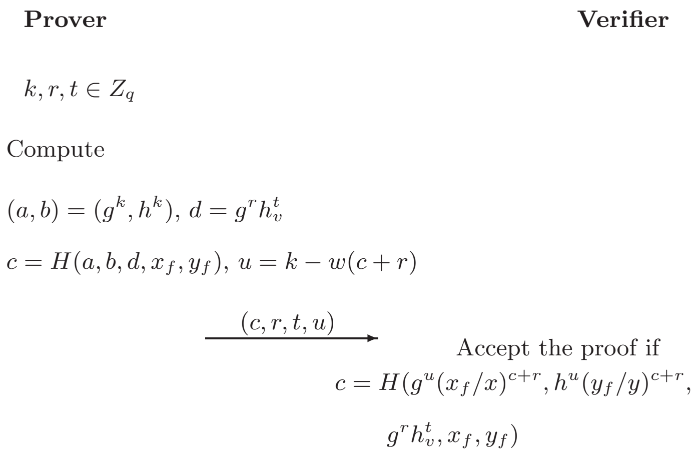

Prover
Verifier
$k, r, t \in Z_q$
Compute
$(a, b) = (g^k, h^k)$, $d = g^r h_v^t$
$c = H(a, b, d, x_f, y_f)$, $u = k - w(c + r)$
(c, r, t, u)
Accept the proof if
$c = H(g^u (x_f/x)^{c+r}, h^u (y_f/y)^{c+r},$
$g^r h_v^t, x_f, y_f)$

Fig. 2. Designated-verifier re-encryption knowledge proof

Appendix B: Designated-Verifier Re-encryption Proof

Let $(x, y) = (g^a, h^a m)$ be an original encrypted ElGamal ciphertext for the message m with a public key $h = g^s$ and $(x_f, y_f) = (xg^w, yh^w)$ be a re-encrypted ciphertext by a prover. The prover wants to prove that x_f/x and y_f/y have the same exponent w without exposing the value of w. Suppose $h_v = g^{s_v}$ be the public key of the verifier.

The verifier can open the commitment d freely with his private key s_v, *i.e.*, he can compute another pair (r', t') such that $r' + s_v t' = r + s_v t$ holds. Therefore, the verifier can generate the re-encryption knowledge proof (c', r', t', u') for any pair (x', y') of his choice, where (α, β, u') are randomly chosen numbers, $c' = H(g^{u'}(x_f/x')^{\alpha}, h^{u'}(y_f/y')^{\alpha}, g^{\delta}, x_f, y_f)$, $r' = \alpha - c'$, $t' = (\delta - r')/s_v$.

Bidder B_i **Seller**

$w_j, d_j, r_j \in_R Z_q$

Compute
$a_{1,j},\ b_{1,j},\ a_{2,j},\ b_{2,j}$

$$\xrightarrow{(a_{1,j}, b_{1,j}), (a_{2,j}, b_{2,j})}$$

$r_{1,j}, r_{2,j}, d_{1,j} \in Z_q$

Compute
$a'_{1,j},\ b'_{1,j},\ a'_{2,j},\ b'_{2,j}$

$$\xleftarrow{(a'_{1,j}, b'_{1,j}), (a'_{2,j}, b'_{2,j})}$$

Compute
$c_j = H(a'_{1,j}, b'_{1,j}, a'_{2,j}, b'_{2,j}), e_j = c_j - d_j, f_j = w_j - a_{i,j} e_j$
$e_{p_i} \leftrightarrow d_{p_i}; f_{p_i} \leftrightarrow r_{p_i}$

$$\xrightarrow{(d_j, r_j, e_j, f_j)}$$

Compute
$d'_{1,j} = d_j + d_{1,j}$
$d'_{2,j} = e_j - d_{1,j}$
$r'_{1,j} = r_j + r_{1,j} - d'_{1,j}\beta_j$
$r'_{2,j} = f_j + r_{2,j} - d'_{2,j}\beta_j$

$$\xleftarrow{(d'_{1,j}, d'_{2,j}, r'_{1,j}, r'_{2,j})}$$

Compute
$c_j = d'_{1,j} + d'_{2,j}$
$$c_j = H(g^{r'_{1,j}}(x^*_{i,j})^{d'_{1,j}}, (h_1h_2)^{r'_{1,j}}(y^*_{i,j}/G_1)^{d'_{1,j}}, g^{r'_{2,j}}(x^*_{i,j})^{d'_{2,j}}, (h_1h_2)^{r'_{2,j}}(y^*_{i,j}/G_2)^{d'_{2,j}})$$

Fig. 3. Proof the validity of the bidding vector

Appendix C: Prove the Validity of the Bidding Vector

Each bidder and the seller jointly generate a proof of the validity of the bidding vector.

In figure 3, if $j = p_i$, let $a_{1,j} = g^{w_j}$, $b_{1,j} = (h_1h_2)^{w_j}$, $a_{2,j} = g^{r_j} x_{i,j}^{d_j}$, $b_{2,j} = (h_1h_2)^{r_2}(y_{i,j}/G_2)^{d_2}$; else, let $a_{1,j} = g^{r_j} x_{i,j}^{d_j}$, $b_{1,j} = (h_1h_2)^{r_j}(y_{i,j}/G_1)^{d_j}$, $a_{2,j} = g^{w_j}$, $b_{2,j} = (h_1h_2)^{w_j}$.

Correspondingly, let $a'_{1,j} = a_{1,j} g^{r_{1,j}} x_{i,j}^{d_{1,j}}$, $b'_{1,j} = b_{1,j}(h_1h_2)^{r_{1,j}}(y_{i,j}/G_1)^{d_{1,j}}$, $a'_{2,j} = a_{2,j} g^{r_{2,j}} x_{i,j}^{-d_{1,j}}$, $b'_{2,j} = b_{2,j}(h_1h_2)^{r_{2,j}}(y_{i,j}/G_2)^{-d_{1,j}}$.

Software Watermarking Through Register Allocation: Implementation, Analysis, and Attacks

Ginger Myles and Christian Collberg

Department of Computer Science
University of Arizona,Tucson, AZ, 85721, USA
{mylesg,collberg}@cs.arizona.edu

Abstract. In this paper we explore the application of the QP watermarking algorithm proposed by G. Qu and M. Potkonjak to software watermarking. The algorithm was originally proposed as a technique for watermarking the graph coloring problem which can be applied to a variety of media such as FPGA designs and software through register allocation. We implemented the algorithm within the SandMark framework, a system that allows the study of watermarking, tamper-proofing, and obfuscation algorithms for Java bytecode. Through the use of this framework we were able to perform an empirical evaluation of the algorithm. In particular we demonstrate that the use of register allocation, while incurring no performance overhead and being stealthy, is in fact vulnerable to attacks such as decompile/recompile. We also demonstrate that the QP algorithm does not allow for accurate watermark recognition without significant modifications.

Keywords. Copyright protection, register allocation, Java bytecode, software piracy, software watermarking.

1 Introduction

It has been estimated that the software piracy business is a $12 billion per year industry [15]. Many different technical means of protecting against piracy have been proposed. Some, such as *dongles* and *tamper-proof processors*, are hardware based. In this paper we discuss a software-based approach, namely Software Watermarking. The idea of software watermarking is based on Media Watermarking which is a well-known technique used to protect images, audio, and video from illegal redistribution. Typically, media watermarking algorithms embed a unique identifier in a multimedia data object by introducing small errors which are not humanly detectable. Since the original functionality of the software must be maintained the introduction of small errors in not a viable option for software watermarking, therefore the algorithms must be based on very different techniques.

The general idea of software watermarking is to embed a message w (the "watermark") into a program P, such that w uniquely identifies the owner of P (w is a copyright notice) or the purchaser of P (w is a fingerprint). A watermarking system consists of two functions $\texttt{embed}(P, w, key) \rightarrow P'$ and $\texttt{recognize}(P', key) \rightarrow w$. The embedding of the message can either be done at the source or binary level. The immediate goal of software watermarking is to be able to prove ownership of the IP, while discouraging copying and illegal distribution of the software. Thus, it is not a technique for preventing piracy, but rather allows the tracing of pirates after the fact.

The QP algorithm proposed by G. Qu and M. Potkonjak [17, 18] is a constraint-based watermarking algorithm that embeds a watermark in a graph through the use of the graph coloring problem. Constraint-based techniques generally start with an optimization problem (usually an NP-complete problem such as graph coloring, scheduling, etc.) that is used to construct the original IP. Then new constraints (representing the watermark or fingerprint) are added to this problem. Solving this new problem will yield a solution that embeds the watermark.

In [13] it is suggested that one possible technique for watermarking software is to modify the register allocation in a unique way. One such technique is the QP algorithm which can be applied to a graph coloring register allocator. In this paper we describe the first known implementation and empirical evaluation of the QP algorithm for software watermarking. We also demonstrate that the QP algorithm contains significant flaws when applied to software watermarking, but can be modified so as to make it a viable algorithm. To date, there have been a variety of techniques proposed for software watermarking, but there has been very little analysis of these techniques. Notable exceptions are Hachez's [11] and Sahoo and Collberg's [21] evaluations of the Stern et. al. algorithm [22].

The goals of this paper are to explore the use of register allocation in software watermarking and to further advance the study of what constitutes a strong watermarking system. We believe that through the implementation and analysis of what are currently theoretical techniques a better understanding of what constitutes a strong watermarking system can be developed, as well as a means of comparing the various techniques. This will hopefully allow us to be able to concluded that certain techniques are better than others.

The remainder of the paper is organized as follows. Section 2 contains a brief description of the variety of techniques that have been proposed for software watermarking. In Sect. 3 we discuss the QP algorithm and why a direct implementation is not possible for software watermarking. Section 4 presents the modifications necessary to improve the strength of the algorithm for software watermarking and in Sect. 5 we provide an empirical evaluation of the algorithm. This is followed by a brief conclusion in Sect. 6.

2 Related Work

There are two major types of software watermarks: static and dynamic. The general idea behind the dynamic watermarking technique is that information from the execution of the program is used to embed the watermark. The technique of dynamic watermarking was first proposed by Collberg and Thomborson [9] in which a watermark is embedded in a graph structure that is created during execution.

The QP algorithm fits into the static watermarking category because the embedding of the watermark does not rely on any runtime information. There are other static watermarking algorithms such as those proposed by Davidson and Myhrvold [10], Venkatesan et al. [23], and Stern et al. [22]. In [23] the authors describe a "graph-theoretic" approach where a watermark is embedded in the program's control-flow graph by adding a new sub-graph which represents the watermark. This algorithm is basically an extension of the algorithm proposed by Davidson and Myhrvold, which embeds a watermark by reordering blocks in an execution flow. Since a watermarking algorithm must preserve the semantics of a program, the blocks which have been reordered are modified so that the original execution flow is maintained. The Stern, et al. algorithm uses a spread spectrum technique where the watermark is embedded throughout the program by modifying the instruction frequencies.

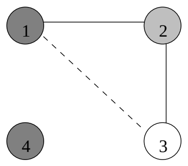
Outside of the SandMark project [4, 8], the only published example of an evaluation of a software watermarking algorithm is that of Hachez [11] which does not provide an evaluation to the extent of that in [21].

3 The QP Algorithm

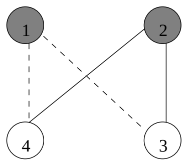
Qu and Potkonjak have proposed several methods for embedding a watermark message in a graph [16, 17, 18, 19, 20, 24]. We have chosen to implement and study the *edge-adding* technique first proposed in [17, 18]. The idea behind QP is that edges are added between chosen vertices in a graph based on the value of the message. Since these vertices are now connected, when the graph is colored the vertices will be colored with different colors.

The message is embedded as follows: Given a graph $G(V, E)$ and a message M, order the vertices V and convert the message to binary ($M = m_0 m_1 \ldots$). For each v_i, find the *nearest* two vertices v_{i_1} and v_{i_2} which are not connected to v_i, where nearest means $i_2 > i_1 > i(mod\ n)$, the edges $(v_i, v_{i_1}), (v_i, v_{i_2}) \notin E$ and $(v_i, v_j) \in E$ for all $i < j < i_1, i_1 < j < i_2$. If $m_i = 0$ add edge (v_i, v_{i_1}), else add edge (v_i, v_{i_2}). The edges that are added represent fake interferences and force a new coloring of the graph.

The message is recognized as follows: Given a graph $G(V, E)$, for each pair of vertices (v_i, v_j), $j > i(mod\ n)$, which are not connected by an edge and are different colors, one bit of the message can be obtained. The bit extraction is done by examining how many vertices occur between v_i and v_j which are not connected to v_i. There are three cases to consider:



(a) Using case 1 a 0 bit can be extracted.



(b) Using case 2 a 1 bit can be extracted and using case 3 a 0 bit can be extracted.

Fig. 1. Illustration of the three cases that can be encountered during recognition

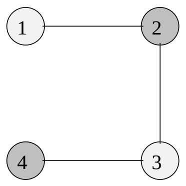
Case 1: If there are no unconnected vertices between v_i and v_j we extract a 0 bit. This case is illustrated in Fig. 1(a) where solid edges represent edges in the graph and dashed edges indicate an edge that was added during embedding. Suppose we perform recognition on the vertices $v_i = 1$ and $v_j = 3$. Since the only vertex between vertices 1 and 3 is connected to vertex 1 we found a 0 bit.

Case 2: If there is one vertex between v_i and v_j which is not connected to v_i then the hidden bit is 1. This case can be illustrated using $v_i = 1$ and $v_j = 3$ in Fig. 1(b). Since vertex 2 lies between vertices 1 and 3 and is not connected to vertex 1 we recognize a 1 bit.

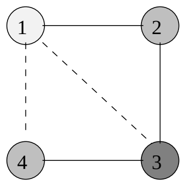
Case 3: If there are two or more vertices not connected to v_i between v_i and v_j then reverse the order of the two vertices and repeat the process. To understand how this case could arise consider the graph in Fig. 1(b). Assume we are performing recognition on the vertices $v_i = 1$ and $v_j = 4$. Since there are two vertices between 1 and 4 that are not connected to 1, it is not possible that 4 was chosen as either v_{i_1} or v_{i_2} during embedding. Instead it must be that 4 was chosen as v_i and 1 was chosen as v_{i_1} which means that a 0 was embedded.

3.1 Recognition Failure

While studying the viability of applying the QP algorithm to software watermarking we discovered that the recognition algorithm was problematic in two different ways. The first flaw involves recognizing the wrong bit, e.g. we recognize a 1 bit when we embedded a 0. To illustrate this error consider the graphs in Fig. 2(a) and 2(b). We wish to embed the message $M = 00$ in the graph in Fig. 2(a). By following the above algorithm bit m_1 is embedded by choosing $v_1 = 1, v_{1_1} = 3, v_{1_2} = 4$ and adding an edge between vertices 1 and 3. Bit m_2



(a) Before



(b) After

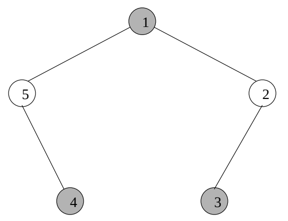
Fig. 2. Recognition failure occurs when the message 01 is recovered from the interference graph even though the message 00 was embedded

is embedded by choosing $v_2 = 4, v_{2_1} = 1, v_{2_2} = 2$ and adding an edge between vertices 4 and 1. The new interference graph as well as the new coloring can be seen in Fig. 2(b). When the QP recognition algorithm is used the first set of vertices selected are 1 and 3. Since there are no vertices between 1 and 3, that are not connected to 1, the recovered bit is a 0. The second set is 1 and 4. There is 1 vertex between 1 and 4 that is not connected to 1, so the recovered bit is 1. Thus, we recognized the message 01 even though the message 00 was embedded. The second flaw is that in many instances we recover more bits than we embedded. For example, when we embed the message 101 in the graph in Fig. 3(a) we obtain the graph in Fig. 3(b). By following the recognition algorithm we recover the message 1001. In fact, we discovered that as the number of vertices in the graph increases so does the number of extra bits recovered.

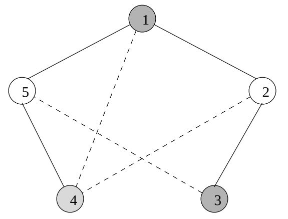
The cause of both of the problems can be traced back to a single assumption made about the coloring of the vertices: any two unconnected vertices are colored differently if and only if an edge is added between them during embedding. This assumption turns out to be incorrect as illustrated in Fig. 3. When we embed the message 101 it forces vertices 3 and 4 to be colored differently even though no edge is added between them. This assumption fails because any particular vertex could be included in multiple *triples*. Thus, even if an edge is not added in one selection it could be in another. The unpredictability is also influenced by vertices in a triple that are not all of the same color. When this occurs it is possible that the color of the vertices could be influenced by the coloring of an adjacent vertex which is not part of the triple.

Definition 1 (triple). *Given a graph $G = (V, E)$, a set of 3 vertices $\{v_1, v_2, v_3\}$ is considered a triple if*

1. $v_1, v_2, v_3 \in V$*, and*
2. $(v_1, v_2), (v_1, v_3), (v_2, v_3) \notin E$



(a) Before



(b) After

Fig. 3. Embedding the message 101 in the original interference graph yields a graph where the recognition algorithm recovers the message 1001

Since accurate recovery is crucial in software watermarking, where the goal is to provide proof of ownership, we are unable to strictly apply the QP algorithm. Instead we take the same general idea of applying additional constraints to the graph based on the watermark, but we apply the constraints in such a way that the watermark is accurately recovered for all applications that have not been tampered with. The details of the algorithm are explained in the next section.

4 QPS Algorithm

We will next describe several improvements to the QP watermark embedding algorithm that make watermark recognition completely reliable baring any tampering. We call this new version *QPS* (*QP for SandMark*). We noticed during the viability study of the QP algorithm that inaccurate message recovery is due to the unpredictability of the coloring of the vertices. To eliminate this unpredictability the QPS embedding algorithm places additional constraints on which vertices can be selected for a triple. We call these triples *colored triples*. The key idea is to select the triples so that they are isolated units that will not effect other vertices in the graph. This is accomplished using the QPS embedding algorithm in Fig. 4(a) where a colored triple is selected and an edge is added based on the bit value. In addition, we use a specially designed register allocator which only changes the coloring of one of the two vertices involved in the added edge and no other vertices in the graph.

Definition 2 (colored triple). *Given an n-colorable graph $G = (V, E)$, a set of three vertices $\{v_1, v_2, v_3\}$ is considered a colored triple if*

1. $v_1, v_2, v_3 \in V$,
2. $(v_1, v_2), (v_1, v_3), (v_2, v_3) \notin E$, *and*
3. v_1, v_2, v_3 *are all colored the same color.*

Theorem 1. *Given an n-colorable graph $G = (V, E)$ that contains a colored triple, adding an edge between 2 vertices in the triple will only alter the coloring of one of those 2 vertices and no other vertices in the graph. In addition, G' is at most $(n+1)$-colorable.*

Corollary 1. *Given an n-colorable graph $G = (V, E)$ which contains k independent colored triples, adding an edge within each of the triples will only alter the coloring of a single vertex in each triple. In addition, G' is at most $(n+k)$-colorable.*

The QPS recognition algorithm is slightly more complex than the QP algorithm and relies on additional information. For this algorithm two colored interference graphs are required, the coloring that was embedded in the method and the original coloring for that method. The original coloring is needed because the recognition basically repeats the embedding. The algorithm works by identifying triples as they would have been selected in the embedding phase. Once a triple (v_i, v_{i_1}, v_{i_2}) has been identified that same triple is examined in the embedded coloring $(v'_i, v'_{i_1}, v'_{i_2})$. If v'_i and v'_{i_1} are different colors then we know that a 0 was embedded, otherwise a 1 was embedded. The QPS recognition algorithm can be seen in Fig. 4(b).

```
for each vertex $v_i \in V$ which is not already in a triple
    if possible find the nearest two
        vertices $v_{i_1}$ and $v_{i_2}$ such that
         $v_{i_1}$ and $v_{i_2}$ are the same color as $v_i$,
        and $v_{i_1}$ and $v_{i_2}$ are not already in a triple.
    if $m_i = 0$
        add edge $(v_i, v_{i_1})$
    else
        add edge $(v_i, v_{i_2})$
end for
```

(a) QPS embedding algorithm

```
for each vertex $v_i \in V$ which is not already in a triple
    if possible find the nearest two
        vertices $v_{i_1}$ and $v_{i_2}$ such that
         $v_{i_1}$ and $v_{i_2}$ are the same color as $v_i$,
        and $v_{i_1}$ and $v_{i_2}$ are not already in a triple.
    if $v'_i$ and $v'_{i_1}$ are different colors
        found a 0
        add edge $(v_i, v_{i_1})$
    else
        found a 1
        add edge $(v_i, v_{i_2})$
end for
```

(b) QPS recognition algorithm

Fig. 4. QPS embedding and recognition algorithms

4.1 Implementation Details

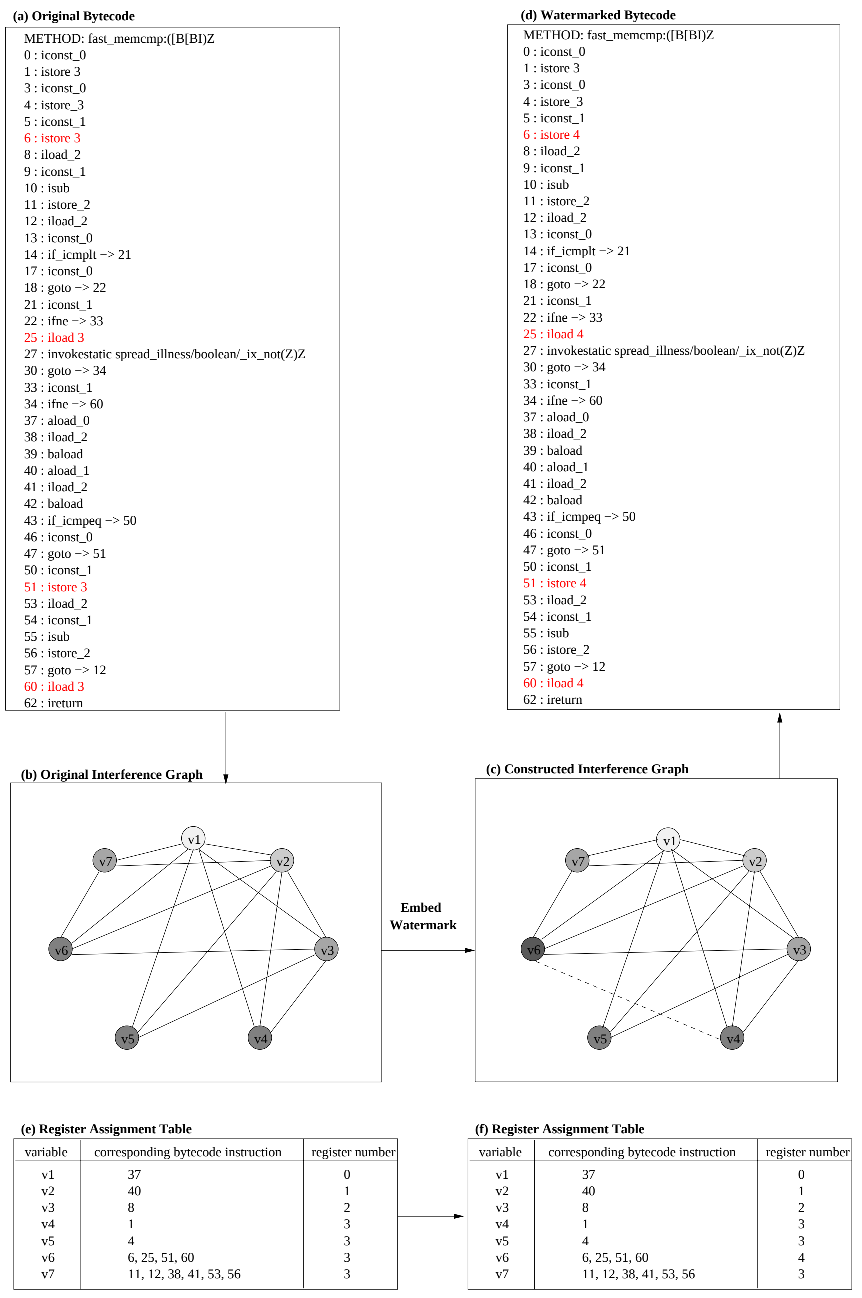
We implemented the QPS algorithm for Java using the BCEL bytecode editor [1] and incorporated it into the SandMark framework. The watermarking algorithm can be applied to an entire application or to a single class file. In either case, the watermark is spread across all methods and is embedded as many times as possible. This increases the likelihood that the message can be recovered in the event of application tampering. Recovering the same message multiple times within an application demonstrates an increased probability that the message was intentionally placed in the application as proof of ownership.

The message is distributed throughout the class files by embedding as many bits of the message as possible in each of the methods. We sort the methods based on their signature and then choose the methods using a pseudo-random number generator seeded with the watermark key. The process is repeated until all methods have been marked. To recover the message we again sort the methods based upon their signatures. The order in which we choose the methods to extract message bits is done using a pseudo-random number generator seeded with the watermark key. This allows us to examine the method in the same order as they were selected for embedding. The process is repeated until we have examined every method in the application.

4.2 Preliminary Example

To help illustrate the general idea behind the QPS algorithm we present a simple example that embeds a single bit, with a value of 1, in a Java method. The embedding process requires the modification of the bytecode, but only those instructions which reference local variables, i.e. `load`, `store`, and `iinc` instructions. To accomplish the embedding a new local variable assignment is imposed on the method through the addition of edges to the interference graph. (In Java registers are referred to as local variables.)

Figure 5 (a) shows the original bytecode for the method `fast_memcmp`. The instructions in this bytecode method that reference local variables are 1, 4, 6, 8, 11, 12, 25, 37, 38, 40, 41, 51, 53, 56, and 60 and thus these instructions could be modified during the embedding process. The first step is to perform liveness analysis on the method bytecode to construct the interference graph in Fig. 5 (b). Each node in the graph represents a set of instructions as well as a local variable assignment. In the figure the local variable assignment is reflected through the coloring of the vertices. An edge in the graph indicates that the variables are live concurrently and cannot be assigned to the same local variable. Figure 5 (e) shows the interference graph node, the corresponding bytecode instructions, and which local variable is used for those instructions. This particular method contains only a single triple, so only a single bit can be embedded. The identified triple consists of vertices `v4`, `v5`, and `v6`. To embed a bit with value 1 an edge is added between `v4` and `v6`. Because of this new interference, `v4` and `v6` can no longer occupy the same local variable, so a new local variable assignment is imposed on the method. The new assignment is


(b) Original Interference Graph
v1
v2
v3
v4
v5
v6
v7
Embed Watermark
(c) Constructed Interference Graph
v1
v2
v3
v4
v5
v6
v7

(a) Original Bytecode

```
METHOD: fast_memcmp:([B[BI)Z
0 : iconst_0
1 : istore 3
3 : iconst_0
4 : istore_3
5 : iconst_1
6 : istore 3
8 : iload_2
9 : iconst_1
10 : isub
11 : istore_2
12 : iload_2
13 : iconst_0
14 : if_icmplt -> 21
17 : iconst_0
18 : goto -> 22
21 : iconst_1
22 : ifne -> 33
25 : iload 3
27 : invokestatic spread_illness/boolean/_ix_not(Z)Z
30 : goto -> 34
33 : iconst_1
34 : ifne -> 60
37 : aload_0
38 : iload_2
39 : baload
40 : aload_1
41 : iload_2
42 : baload
43 : if_icmpeq -> 50
46 : iconst_0
47 : goto -> 51
50 : iconst_1
51 : istore 3
53 : iload_2
54 : iconst_1
55 : isub
56 : istore_2
57 : goto -> 12
60 : iload 3
62 : ireturn
```

(d) Watermarked Bytecode

```
METHOD: fast_memcmp:([B[BI)Z
0 : iconst_0
1 : istore 3
3 : iconst_0
4 : istore_3
5 : iconst_1
6 : istore 4
8 : iload_2
9 : iconst_1
10 : isub
11 : istore_2
12 : iload_2
13 : iconst_0
14 : if_icmplt -> 21
17 : iconst_0
18 : goto -> 22
21 : iconst_1
22 : ifne -> 33
25 : iload 4
27 : invokestatic spread_illness/boolean/_ix_not(Z)Z
30 : goto -> 34
33 : iconst_1
34 : ifne -> 60
37 : aload_0
38 : iload_2
39 : baload
40 : aload_1
41 : iload_2
42 : baload
43 : if_icmpeq -> 50
46 : iconst_0
47 : goto -> 51
50 : iconst_1
51 : istore 4
53 : iload_2
54 : iconst_1
55 : isub
56 : istore_2
57 : goto -> 12
60 : iload 4
62 : ireturn
```

(e) Register Assignment Table

variable	corresponding bytecode instruction	register number
v1	37	0
v2	40	1
v3	8	2
v4	1	3
v5	4	3
v6	6, 25, 51, 60	3
v7	11, 12, 38, 41, 53, 56	3

(f) Register Assignment Table

variable	corresponding bytecode instruction	register number
v1	37	0
v2	40	1
v3	8	2
v4	1	3
v5	4	3
v6	6, 25, 51, 60	4
v7	11, 12, 38, 41, 53, 56	3

Fig. 5. Example of using the QPS algorithm to embed a single bit in a method. The process begins with the method's bytecode. An interference graph is constructed from this method using liveness analysis. The interference graph is then modified based on the watermark which forces a new local variable assignment for the method

reflected in Fig. 5 (c) and (f). The result is new bytecode (shown in Fig. 5 (d)) where instructions 6, 25, 51, and 60 now reference local variable 4 instead of 3.

4.3 Algorithm Overview

Next we describe the details of the embedding and recognition algorithms. They are illustrated in Figs. 6 and 7.

Embedding. The first step in the embedding process is to convert the watermark message to binary and add an 8-bit length field. The length field represents the length of the binary message and is used during recognition.

8 bits	0 − 256 bits
length	message

Next, a control flow graph is constructed for the method, and liveness analysis and interference graph construction are performed. Our recognition procedure requires that a base local variable assignment be established prior to adding the fake interferences. We therefore perform an initial register allocation. We then add the fake interferences and a final register allocation is performed. The local variable assignment obtained from this second register allocation is the one which embeds the watermark in the method.

To recognize a watermark, we must establish a unique labeling of the interference graph, i.e., the nodes must be ordered. In our implementation the variables are ordered based on the location of their live ranges in the code. A variable v_1 is considered to be before a variable v_2 if the start of its live range is earlier in the code than the start of v_2's range. If the start of the ranges are the same (as is the case for method parameters), then the end (or first break) in the live-ranges are used.

```
convert the message to binary
pad the front of the watermark with 8 bits that represent the
    length of the binary message
sort all of the methods
seed a pseudo-random number generator with the watermark key
repeat until all methods have been considered
    randomly choose a method
    build CFG
    do liveness analysis
    build interference graph
    do register allocation
    embed bits of the watermark by adding
        fake interference to the interference graph
    do register allocation
end repeat
```

Fig. 6. General QPS embedding algorithm

```
seed a pseudo-random number generator with the key
repeat until each method has been considered
    randomly choose a method
    build CFG
    do liveness analysis
    build interference graph to obtain
        watermarked coloring
    do register allocation
    build interference graph to obtain
        original coloring
    compare colorings to recover bits of
        the embedded message
end repeat
check the length of the recovered binary number and
    take appropriate action
```

Fig. 7. General QPS recognition algorithm

Recognition. The recognition process is repeated for each method in the class. The first step is to build the control flow graph for the method. Next, liveness analysis is performed and an interference graph is constructed. The interference graph contains the local variable assignments that correspond to a sub-string of the embedded message. Once the watermarked coloring has been obtained, the register allocator is run on the interference graph. This step yields the original local variable assignment that existed before the fake interferences were added. Now we have obtained two interference graphs associated with this method: the original one and the one with fake interferences. The two graphs are compared to obtain bits of the message using a process similar to that used when adding the fake interferences.

Once all of the methods have been analyzed and all bits extracted, the length field is used to determine which bits comprise the watermark. This step is necessary when we have attempted to embed multiple copies of the message. For example, say we embed a message with binary length 8. If we had recovered the binary message 0110100101 we would know to throw away the extra two bits since this message has length 10. If we had recovered the binary message 0110100101101001 which has length 16 then we would know that we have recovered two copies of the message. Once all binary copies of the message have been recovered we convert the message back to its ASCII equivalent.

5 Evaluation

We have performed a variety of empirical tests on the QPS watermarking algorithm to evaluate its overall effectiveness. This analysis is explained in this section. There are two significant advantages to implementing the QPS algorithm within the SandMark framework: (1) we are able to perform a variety of attacks on a program that has been watermarked using this algorithm and (2) there are other watermarking algorithms which can be used for comparison purposes. Within SandMark we are able to easily perform attacks such as manual attacks which could involve sorting the methods or analyzing the statistics. We are also able to apply obfuscations to the watermarked code which are semantics-preserving distortive attacks that could destroy the watermark.

Very little work has been done towards the evaluation of strength in watermarking. What we do know is that a watermark needs to be resilient against determined attempts at discovery and removal. In [9] the strength of a watermarking system is based on three properties: *stealth, data-rate, and resilience.* A stealthy watermark is difficult to detect since it exhibits the same properties as the code that surrounds it. An algorithm should also have a high data-rate so that it is possible to store a large secret message. Finally, a watermark needs to be resilient against attacks. There are three important categories of attacks:

– In a *subtractive attack* the adversary examines the (disassembled/de-compiled) program in an attempt to discover the watermark and to remove all or part of it from the code. This must be done in such a way that the functionality of the software is not destroyed.
– In an *additive attack* the adversary adds a new watermark in order to make it hard for the IP owner to prove that her watermark is actually the original.
– In a *distortive attack* a series of semantics-preserving transformations are applied to the software in an attempt to render the watermark useless. It is the goal of the attacker to distort the software in such a way that the watermark becomes unrecoverable but the software's functionality and performance remain intact.

In addition to stealth, data rate, and resilience, Qu and Potkonjak [18] propose the following desirable properties for a software watermark:

– *high credibility*: The watermark should be readily detectable for proof of authorship while minimizing the probability of coincidence. The chance that a watermark is recovered from the program even though it was not embedded should remain low.
– *low overhead*: The degradation of the software by embedding the watermark should be minimized.
– *part protection*: Ideally, a good watermark should be distributed throughout the software or design in order to protect all parts of it.

To evaluate QPS we examined these six software watermark properties which yield a strong evaluation of the overall effectiveness. In order to study these properties we applied QPS to ten applications. Characteristics of nine of these applications are presented in Table 1. The last application is specjvm [5].

5.1 Credibility

The credibility of a watermarking algorithm is contingent on how detectable the watermark is for proof of ownership. As was argued in Sect. 4 the final implementation provides very high credibility. Unless the watermarked application is tampered with the correct message will always be recovered. There is also a very

Table 1. Benchmark applications used in the evaluation of the QPS watermarking algorithm

Application	number classes	number methods	max method size	min method size	total method size
fft	1	10	253	6	603
lu	1	7	162	6	421
probe	1	7	197	6	439
TTT	12	50	116	5	1208
decode	4	20	719	5	1586
illness	16	37	1679	5	3842
machineSim	12	110	602	0	2979
matrix	2	10	270	8	549
puzzle	3	20	466	6	3136

low probability that the message recovered was recovered by coincidence due to the tight constraints applied to choosing the triples during embedding and recognition. This is a definite improvement over the strict implementation of the QP embedding and recognition algorithms. With the QP algorithm there are many situations where it is unclear whether a 0 or 1 was embedded in the graph. This leaves loopholes in the proof of ownership.

5.2 Data-Rate

Through analysis we discovered that the QPS algorithm has a very low data-rate. So low in fact that we were unable to embed even a single character in 9 out of our 10 benchmark applications, which varied in both size and complexity. Table 1 presents a sampling of these programs and their characteristics.

What we discovered is that the more complex the method the fewer bits we are able to embed. This is because the complex methods also have very complex graphs with many interferences. This drastically decreases the likelihood of obtaining three non-interfering nodes, let alone several of these triples.

We did make one attempt at increasing the data rate by performing static inlining. Prior to applying the QPS algorithm an inliner was run on the program which inlined all static methods. The data-rate obtained both with and without inlining can be seen in Table 2. The inlining did increase the data-rate for most of the benchmarks, but it only increased one of them enough to allow for the embedding of at least a single character.

5.3 Resilience

Unfortunately, the QPS algorithm is not resilient to some rather simple attacks. It also does not allow for the embedding of multiple watermarks using the QPS algorithm. Below, we will give examples of subtractive, additive, and distortive attacks that thwart recognition.

Table 2. Results from applying the watermark to the benchmarks both with and without method inlining. Time measures are in seconds

(a) Results from applying the watermark to the benchmarks without method inlining

Application	**number bits embedded**	**time to embed**	**time to recognize**
fft	3	6.803	4.559
lu	3	13.337	11.257
probe	5	11.73	9.8
TTT	0	6.191	4.134
decode	3	66.777	61.103
illness	10	15.118	11.486
machineSim	4	32.648	29.0
matrix	3	24.406	21.804
puzzle	7	1349.545	1475.641
specjvm	342	22533.398	22060.443

(b) Results from applying the watermark to the benchmarks with method inlining

Application	**number bits embedded**	**time to embed**	**time to recognize**
fft	10	557.31	551.023
lu	7	333.668	326.502
probe	7	96.604	94.619
TTT	0	5.951	4. 095
decode	4	2051.639	2213.981
illness	33	15.221	12.428
machineSim	4	33.158	30.701
matrix	3	25.463	23.757
puzzle	12	15671.56	15077.383

Subtractive Attacks. One of the first things that an adversary may do when trying to eliminate a watermark is to decompile the executable. Once the code is decompiled it can be examined to search for obvious watermarks such as dummy fields or methods. If nothing is found he could just throw out the decompiled code and continue running the watermarked version. But he could also recompile the decompiled version, which would have the effect of reassigning the local variables and therefore erasing any portion of the watermark which was embedded via local variable assignment. We performed this test on the inlined, watermarked version of the illness application using the Jad [3] decompiler. This exercise did have the effect of reassigning the local variables and therefore wiping out the watermark.

Table 3. Results from applying a second watermarking technique to the watermarked application illness

Watermark	**illness**
Bogus Expression	+
HatTrick	+
Bogus Initializer	+
SHKQ	-
Add Method Field	-
Bogus Switch	+
Method Renamer	+

+ : *Original watermark found*
- : *Original watermark destroyed*

Additive Attacks. In an additive attack a second watermark is added to the software so that it is difficult to prove which watermark is the original. With the QPS algorithm if a second watermark is embedded using the same technique it will completely wipe out the original. This is because new local variable assignments are applied to the application.

Often it is the case that multiple watermarking techniques are applied to the same application so as to further guarantee the survival of the watermark. When multiple watermarks are applied it is desirable that the second watermark does not wipe out the first. In Table 3 it can be seen that the QPS watermarking algorithm is quite resilient to the embedding of additional watermarks using a different watermarking algorithm. For this particular test case only the `Add Method Field` and the `SHKQ` (Stern et. al. algorithm) watermarking algorithms had the effect of destroying the original watermark.

Distortive Attacks. The watermark is also susceptible to destruction by an obfuscator which adds code to a method. This obfuscation changes the interference graph which could possibly change the vertices selected for a triple.

Obfuscations that add new methods to a class would also be able to destroy the watermark. This type of obfuscation could have the effect of adding bogus bits to the recovered watermark yielding an incorrect watermark. There is the possibility that no bits get added if no triples can be found in the method's interference graph.

SandMark includes a variety of obfuscators. We applied all of them to the inlined version of the application illness as well as the BLOAT Optimizer [2]. This application was chosen because we are able to embed at least one character and recognition of the watermark can be done in a reasonable amount of time. What we can see in Table 4 is that any obfuscation that modifies the method bodies will destroy the watermark and that there are a variety of different techniques for doing this.

Table 4. Results from applying various obfuscations and the BLOAT optimizer to the inlined version of the application illness

Obfuscation	**illness**
Variable Reassigner	-
Degrade	+
Bogus Predicates	-
Name Obfuscator	+
Method Merger	-
Static Method Bodies	-
Thread Contention	+
Publicizer	+
Parameter Reorderer	+
Signature Bludgeoner	-
Set Fields Public	+
Local Variable Reorderer	+
Buggy Code	+
Append Bogus Code	-
Variable Splitter	-
Class Splitter	+
Add Bogus Fields	+
Mofidy If Else	+
Int Array Splitter	-
False Refactor	-
Inliner	-
Block Marker	+
Promote Locals	-
Bloat Optimizer	-

+ : *Watermark found*
- : *Watermark destroyed*

5.4 Perceptual Invisibility (Stealth)

A watermark embedded using local variable assignment is extremely hard to detect since no additional code is added to the methods. The only instructions that are effected are load and stores and it is only the indices which change. Since it would require extensive analysis to discover that the local variable assignment is artificially imposed, a watermark message embedded with the QPS algorithm is quite stealthy.

In addition to the attacks described in Sect. 5.3, an adversary may try to perform a manual attack. In a manual attack the adversary could examine the code or calculate various statistics on the code. The SandMark framework includes a statistics module that calculates a variety of static statistics, e.g. number of methods, number of public methods, number of static fields, number of conditional statements, etc. Since the QPS algorithm only modifies the local variable assignment, the static statistics calculated in SandMark remain unchanged when a program is watermarked using the QPS algorithm with the exception of the scalars. The scalars statistic measures non-array variables and this includes the local variables. A decrease in the scalar variables is noticed because one of the first steps in the QPS algorithm is to run a register allocator over the method. In most cases this has the effect of decreasing the number of local variables used by the method. This change in statistics could cause an adversary to perform a more detailed manual study, but we still believe the change in local variable assignment will be hard to detect and therefore this algorithm displays a great deal of stealth.

5.5 Part Protection

The best watermarking algorithms are those which spread the message throughout the entire software. This has the effect of decreasing the probability that all or part of the message is destroyed. That is exactly what the QPS algorithm does. We embed as many bits of the watermark as possible in each method. Once the watermark has been embedded, if there are still non-watermarked methods remaining we embed the watermark again. The only drawback is if tampering occurs in the first eight bits which represent the length of the watermark. If this happens the conversion from binary to ASCII will be incorrect.

5.6 Overhead

There are two types of overhead to consider. The first is the overhead associated with actually embedding a watermark in the application and the second is the overhead the application incurs from being watermarked. In the first case what we see in Table 2 is that the running time of the algorithm is quite slow. We have determined that the performance issues can be attributed to: (1) we attempt to embed a portion of the watermark in every method in the application and (2) the analysis required to build the interference graph and color it is quite slow.

The register allocator that is used in the QPS algorithm is a slightly more optimized version than that used by either `javac` or `jikes` which are the most common Java compilers. Because of this the QPS algorithm actually decreases the number of bytes in the application once it has been watermarked. For example, the inlined version of illness is 6143 bytes and once it is watermarked it is 6038 bytes.

6 Conclusion

In [13, 17, 18] Qu and Potkonjak suggest an algorithm for software watermark embedding. We have shown that this algorithm is seriously flawed, in that it does not permit reliable recognition. In addition, the algorithm was not previously submitted to an empirical evaluation of the bit-rate and robustness of the embedding. We have proposed a new version of QP, QPS, which allows robust recognition of watermarks in the absence of attacks. However, our extensive empirical evaluation of QPS shows that it (and hence QP) is susceptible to a large number of distortive attacks. Furthermore, the algorithm has a very low bit-rate.

Our conclusion is that the QP algorithm is unsuitable for software watermarking of architecture-neutral code (such as Java bytecode) in the presence of a determined attacker. However, it is conceivable, that the QPS algorithm could be useful to watermark *native* code where temper-proofing (such as suggested by [6, 7, 12, 14]) might be used to prevent the code from being manipulated. Unfortunately, such tamper-proofing typically adds a fair amount of overhead. In future research we will examine whether tamper-proofed QPS-watermarked native code is resilient to attacks, what the resulting overhead might be, and whether native code will have more opportunities for embedding large watermarks.

Acknowledgments

This work is supported by the NSF under grant CCR-0073483, by the AFRL under contract F33615-02-C-1146 and the GAANN Fellowship.

References

[1] Bcel. `http://jakarta.apache.org/bcel/`. 281
[2] Bloat: The bytecode-level optimizer and analysis tool. `http://www.cs.purdue.edu/s3/projects/bloat/`. 288
[3] Jad - the fast java decompiler. `http://www.geocities.com/zz_xu/jad.html`. 287
[4] Sandmark. `http://www.cs.arizona.edu/sandmark/`. 276
[5] Spec jvm98 v1.04. `http://www.specbench.org/osg/jvm98/`. 285
[6] David Aucsmith. Tamper resistant software: An implementation. In Ross J. Anderson, editor, *Information Hiding, First International Workshop*, pages 317–333, Cambridge, U.K., May 1996. Springer-Verlag. Lecture Notes in Computer Science, Vol. 1174. 291
[7] H. Chang and Mike Atallah. Protecting software code by guards. In *Workshop on Security and Privacy in Digital Rights Management 2001*, 2002. `http://www.star-lab.com/sander/spdrm/papers.html`. 291
[8] Christian Collberg, Ginger Myles, and Andrew Huntwork. Sandmark - a tool for software protection research. *IEEE Security and Privacy*, 1(4):40–49, 2003. 276
[9] Christian Collberg and Clark Thomborson. Software watermarking: Models and dynamic embeddings. In *Principles of Programming Languages 1999, POPL'99*, pages 311–324, 1999. 276, 285
[10] R. L. Davidson and N. Myhrvold. Method and system for generating and auditing a signature for a computer program. US Patent 5,559,884, Assignee: Microsoft Corporation, 1996. `http://www.delphion.com/details?pn=US05559884__`. 276
[11] Gael Hachez. *A Comparative Study of Software Protection Tools Suited for E-Commerce with Contributions to Software Watermarking and Smart Cards.* PhD thesis, Universite Catholique de Louvain, 2003. 275, 276
[12] Bill Horne, Lesley Matheson, Casey Sheehan, , and Robert E. Tarjan. Dynamic self-checking techniques for improved tamper resistance. In *Security and Privacy in Digital Rights Management, ACM CCS-8 Workshop DRM 2001*, Philadelphia, PA, USA, November 2001. Springer Verlag, LNCS 2320. 291
[13] F. Koushanfar, G. Qu, and M. Potkonjak. Intellectual property metering. In *4th Information Hiding Workshop*, pages 87–102, 2001. 275, 290

[14] C. Linn and S. K. Debray. Obfuscation of executable code to improve resistance to static disassembly. In *ACM Conference on Computer and Communications Security*, October 2003. 291
[15] International Planning and Research Corporation. Sixth annual BSA global software piracy study, 2001. 274
[16] G. Qu, J.L. Wong, and M. Potkonjak. Fair watermarking techniques. In *EEE/ACM Asia and South Pacific Design Automation Conference*, pages 55–60, 2000. 276
[17] Gang Qu and Miodrag Potkonjak. Analysis of watermarking techniques for graph coloring problem. In *ICCAD*, pages 190–193, 1998. 275, 276, 290
[18] Gang Qu and Miodrag Potkonjak. Hiding signatures in graph coloring solutions. In *Information Hiding*, pages 348–367, 1999. 275, 276, 285, 290
[19] Gang Qu and Miodrag Potkonjak. Fingerprinting intellectual property using constraint-addition. In *Design Automation Conference*, pages 587–592, 2000. 276
[20] Gang Qu, Jennifer L. Wong, and Miodrag Potkonjak. Optimization-intensive watermarking techniques for decision problems. In *Design Automation Conference*, pages 33–36, 1999. 276
[21] Tapas Ranjan Sahoo and Christian Collberg. Software watermarking in the frequency domain: Implementation, analysis, and attacks. In *ACM Symposium on Applied Computing*, March 2004 (to appear). 275, 276
[22] Julien P. Stern, Gael Hachez, Francois Koeune, and Jean-Jacques Quisquater. Robust object watermarking: Application to code. In *Information Hiding*, pages 368–378, 1999. http://citeseer.nj.nec.com/stern00robust.html. 275, 276
[23] Ramarathnam Venkatesan, Vijay Vazirani, and Saurabh Sinha. A graph theoretic approach to software watermarking. In *4th International Information Hiding Workshop*, Pittsburgh, PA, April 2001. http://link.springer.de/link/service/series/0558/bibs/2137/21370157.htm. 276
[24] Gregory Wolfe, Jennifer L. Wong, and Miodrag Potkonjak. Watermarking graph partitioning solutions. In *Design Automation Conference*, pages 486–489, 2001. 276

A Proofs

Notation:

1. $\chi(v_i)$: the color of the vertex v_i
2. $\chi(V)$: $\bigcup_{v_i \in V} \chi(v_i)$
3. $|\chi(V)|$: $\left|\bigcup_{v_i \in V} \chi(v_i)\right|$

Theorem 1. *Given an n-colorable graph $G = (V, E)$ that contains a colored triple, adding an edge between 2 vertices in the triple will only alter the coloring of one of those 2 vertices and no other vertices in the graph. In addition, G' is at most $(n + 1)$-colorable.*

Proof. Let $G = (V, E)$ be a graph such that $|\chi(V)| = n$ and there exists at least one colored triple $\{v_1, v_2, v_3\}$. Let $\chi(v_1), \chi(v_2), \chi(v_3) = n$. There are 3 possibilities for adding an edge within $\{v_1, v_2, v_3\}$. So without loss of generality, add edge (v_1, v_2). In the general case suppose v_1 is adjacent to a set of i vertices V_i and that v_2 is adjacent to a set of j vertices V_j where $V_i \cup V_j \geq 0$. We know that $\chi(v_1) \notin \chi(V_i)$ and $\chi(v_2) \notin \chi(V_j)$. Thus to color $G' = (V, E \cup (v_1, v_2))$ leave all $v_k \in V$ their original color except alter $\chi(v_2)$ such that $\chi(v_2) \notin (\chi(V_j) \cup \chi(v_1))$. In the worst case, v_1 is adjacent to $n - 1$ vertices, V_{l_1}, such that $|\chi(V_{n-1})| = n - 1$ and v_2 is adjacent to $n - 1$ vertices, V_{l_2}, such that $|\chi(V_{n-1})| = n - 1$ where $V_{l_1} \cup V_{l_2} \geq 0$. In this case, to color G' leave all $v_k \in V$ their original color except alter $\chi(v_2)$. In this case $\chi(v_2) = n+1$. This coloring only increases $|\chi(V)|$ by 1. □

Corollary 1. *Given an n-colorable graph $G = (V, E)$ which contains k independent colored triples, adding an edge within each of the triples will only alter the coloring of a single vertex in each triple. In addition, G' is at most $(n + k)$-colorable.*

Proof. This proof follows directly from the proof of Theorem 1. □

Analysis of the Bounds for Linear Block Codes in Watermark Channel

Limin Gu, Jiwu Huang[⋆], and Zewen Chen

Dept. of Electronics,
Sun Yat-Sen University, Guangzhou 510275, P. R. China.
isshjw@zsu.edu.cn

Abstract. Robustness is one of the fundamental requirements of digital watermarking in many applications. To improve watermark robustness, some researchers proposed to employ error correcting coding (ECC). However, due to the constraint of imperceptibility, the redundancy introduced by ECC will decrease embedding strength of the watermark. That is, coding ratio of ECC greatly affects the amplitude of watermark signal. It is this point that makes watermark channel different from conventional communication channel. Thus, a question arises. How to choose the coding ratio so that ECC could bring coding gain in the watermark channel? This paper analytically and experimentally investigates the problem. By using Plotkin Bound and Hamming Bound, we present the coding ratio constraints of binary and non-binary linear block codes. Our conclusions can easily extend to convolutional codes.

1 Introduction

Due to wide potential applications, information hiding and digital watermarking techniques have attracted great attention in recent years [1, 2]. Imperceptibility and robustness are two fundamental but contradictive requirements in watermarking. One goal of watermarking techniques is to achieve the best robustness under the constraints of imperceptibility.

Owing to the similarities between watermark channel and digital communication channel, watermark channel was viewed as a digital communication problem [3]. Since ECC is effectively used in noisy channel in digital communications, it seems to be straightforward to apply ECC in watermark channel. Some algorithms applying ECC to improve the robustness of watermarking have been reported [4-6].

However, it is noted that there exist some important differences between watermark channel and conventional communication channel. The former is under the constraint of imperceptibility. Thus, several problems arise as follows.

[⋆] contacting author

(1) Under what condition could ECC have coding gain in the watermark channel?
(2) Which code is the best choice in watermark channel?
(3) How to choose the coding ratio?

Some studies relevant to above issues have been conducted. Huang et al. [7] compared the performance of repetition code, BCH code and correlation detection. Baudry et al. [8] investigated ECC strategies in watermarking, and analyzed the performance of BCH codes, repetition codes, and their concatenations. Balado et al. [9] claimed that Turbo coding schemes have lower error rate than hybrid coding for embedding the same amount of watermark information bits. Desset et al. [10] proposed the theoretical channel bit error rate (BER) interval, on which repetition code outperforms BCH code. But the problem of how to choose coding ratio remains not solved so far.

In this paper, we address this issue. Because of the constraint of imperceptibility, there is a trade-off between payload (hidden information bits) and watermark embedding strength. Hence, the redundancy introduced by ECC will lead to a decrease of embedding strength. That is, coding ratio of ECC greatly affects the amplitude of watermark signal. It is this point that makes watermark channel different from conventional communication channel. Based on this knowledge, this paper analytically and experimentally investigates the problem. By using Plotkin Bound and Hamming Bound, we present the coding ratio constraints of binary and non-binary linear block codes, and our conclusions can readily extend to convolutional codes.

This paper is organized as follows. Section 2 investigates the performance of ECC in watermark channel, by making use of the differences between watermark channel and conventional communication channel. Based on Plotkin Bound and Hamming Bound, the bounds of coding ratio in watermark channel are derived in this part. In Section 3, we present the simulation results. Finally, conclusions are drawn in Section 4.

2 Performance of ECC in Watermark Channel

In watermark channel, there is a trade-off between the payload and the embedding strength. That is, an increase of hidden bits will result in a decrease of the embedding strength, and vice versus. This trade-off makes watermark channel different from conventional communication channel. It is this difference that motivates our investigation reported in this paper.

2.1 Constraints for Coding Ratio

In watermark embedding, quantization-based model is frequently used. So, we adopted (1) [11] as the embedding model, and concerned with unitary transform domain.

$$\begin{cases} y_k^{'} = y_k - y_k \bmod S + \frac{3}{4}S, & \text{if } \ w_k = 1, \\ y_k^{'} = y_k - y_k \bmod S + \frac{1}{4}S, & \text{if } \ w_k = 0. \end{cases} \tag{1}$$

where y_k and $y_k^{'}$ are the transform domain coefficients of the cover image and its modified version. S is the embedding strength. w_k denotes a hidden watermark bit. The operator *mod* calculates the modulus. Without loss of generality, we assume that the channel noise is additive Gaussian noise with zero mean and variance σ^2. According to [12], the channel bit error rate (BER) p_e^u can be expressed as

$$p_e^u = Q(\frac{S}{4\sigma}) = Q(x) \tag{2}$$

where S represents the embedding strength with a payload of L_1 bits and given PSNR, T_{PSNR}. We can observe that the channel BER is related to watermark embedding strength, if keeping the noise unchanged.

Under the constraint of imperceptivity for watermark channel (keeping the PSNR unchanged), the embedding strength, S, is directly correlated to payload, K. Let K_1 and K_2, S_1 and S_2 denote the payloads and embedding strengths in two different coding schemes, respectively. If keeping the host signal, transform domain, and embedding model unchanged, there will be [13]

$$S_2 = S_1 * \sqrt{\frac{K_1}{K_2}} \tag{3}$$

In watermarking scheme using ECC, we consider the use of non-binary linear block code *[n, k, d]* on *GF(q)*. Usually, $q = 2^m$, such that m information bits are mapped into one of the q symbols. In this case, the encoding procedure maps $k\times 2^m$ binary bits into $n\times 2^m$ binary bits. So, we still use (1) as the embedding model. In this case, L_1 bits watermark message is mapped into L_2 bits code words to be embedded. Keeping the PSNR, T_{PSNR}, and taking into account of (2) and (3), the channel symbol error rate p_e^c of q-symbol block codes is

$$p_e^c = Q(\frac{S}{4\sigma}\sqrt{\frac{L_1}{L_2}}) = Q(x\sqrt{r}) \tag{4}$$

where coding ratio $r = L_1/L_2$. Please note that an increase of coding ratio r results in a decrease of channel symbol error rate, and vice versus. The symbol error rate after decoding p_{sig}^c[12] is

$$p_{sig}^c = \sum_{i=t+1}^{n} \binom{n}{i} (p_e^c)^i (1-p_e^c)^{n-i} \tag{5}$$

where $t = (d-1)/2$ represents the correcting ability of the code *[n, k, d]*. Obviously, $p_{sig}^c$is a binomial variable, and can be well approximated by a Gaussian variable for moderate values of n. Then, we have

$$p_{sig}^c = Q(\frac{t+1-np_e^c}{\sqrt{np_e^c(1-p_e^c)}}) \tag{6}$$

Hence, whether ECC has coding gain is equal to evaluate whether following inequality is satisfied:

$$p_{sig}^c - p_e^u \leq 0 \tag{7}$$

If above inequality is satisfied, ECC could bring coding gain for watermarking. Otherwise, there would be no coding gain. In addition, by considering (2) and (6), we have

$$Q(\frac{t+1-np_e^c}{\sqrt{np_e^c(1-p_e^c)}}) - Q(x) \leq 0 \tag{8}$$

Because $Q-$function is monotone decreasing, we have

$$t+1-np_e^c \geq x\sqrt{np_e^c(1-p_e^c)} \tag{9}$$

$$\begin{cases} (n^2+x^2n)p_e^{c^2} - [2(t+1)n+x^2n]p_e^c + (t+1)^2 \geq 0 \\ t+1-np_e^c \geq 0 \end{cases} \tag{10}$$

Then, we have

$$\begin{cases} p_e^c \leq p_1 \\ p_e^c \leq \frac{t+1}{n} \end{cases} \text{or} \begin{cases} p_e^c \geq p_2 \\ p_e^c \leq \frac{t+1}{n} \end{cases} \tag{11}$$

where

$$\begin{array}{c} p_1 = \frac{[2(t+1)n+nx^2]-\sqrt{\delta}}{2(n^2+nx^2)}, \\ p_2 = \frac{[2(t+1)n+nx^2]+\sqrt{\delta}}{2(n^2+nx^2)}, \\ \delta = n^2x^4 + 4n(t+1)[n-(t+1)]x^2 \end{array} \tag{12}$$

Then, we simplify Inequalities 11. With regard to block codes *[n, k, d]*, $d = 2t+1$, the minimum Hamming distance d satisfies $d \leq n$. Hence, we have $t+1 < n$.

1. Let us prove $p_2 > \frac{t+1}{n}$.

$$\begin{aligned} p_2 - \frac{t+1}{n} &= \frac{[2(t+1)n+nx^2]+\sqrt{\delta}-2(t+1)(n+x^2)}{2(n^2+nx^2)} \\ &= \frac{nx^2+\sqrt{\delta}-2(t+1)x^2}{2(n^2+nx^2)} \end{aligned} \tag{13}$$

Considering $t+1 < n$, we have $\sqrt{\delta} > nx^2$. Equality (13) evolves into

$$p_2 - \frac{t+1}{n} > \frac{2nx^2 - 2(t+1)x^2}{2(n^2+x^2n)} = \frac{[n-(t+1)]x^2}{(n^2+x^2n)} > 0 \tag{14}$$

2. Similarly, we can prove $p_1 < \frac{t+1}{n}$.

So, Inequalities (11) eventually reduces to $p_e^c \leq p_1$. Then, we have

$$Q(x\sqrt{r}) \leq \frac{[2(t+1)n+nx^2]-\sqrt{\delta}}{2(n^2+nx^2)} \tag{15}$$

$$r \geq \left(\frac{1}{x}Q^{-1}(\frac{[2(t+1)n+nx^2]-\sqrt{\delta}}{2(n^2+nx^2)})\right)^2 \tag{16}$$

where $Q^{-1}(\bullet)$ is the inverse function of Q-function.

So, Inequality (16) gives the lower bound of coding ratio, r. That is, the coding ratio cannot be arbitrarily low. This represents the differences between

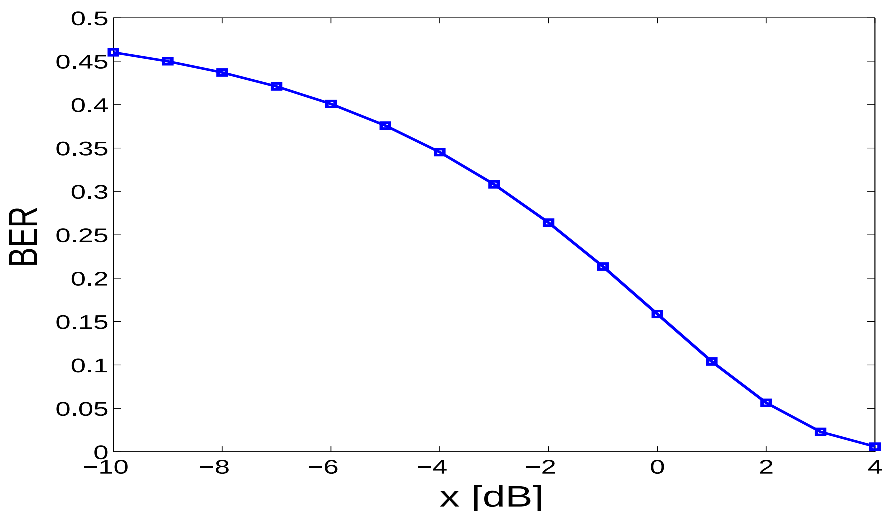


Fig. 1. Distribution of channel BER p_e^u

watermark channel and conventional communication channel. If correcting ability, t, is available when introducing block code to watermarking, using above inequality, it is easy to determine whether block code is profitable. If t is not available, some kind of estimation of t is indispensable.

Then, we investigate the relationship between the bound derived from (16) and $(t+1)$. Considering the following equality

$$\begin{aligned}\frac{\partial p_1}{\partial(t+1)} &= \frac{4n\sqrt{\delta}-4n^2x^2+8n(t+1)x^2}{4(n^2+nx^2)\sqrt{\delta}} \\ &= \frac{4n(\sqrt{\delta}-nx^2)+8n(t+1)x^2}{4(n^2+nx^2)\sqrt{\delta}}\end{aligned} \tag{17}$$

and $\sqrt{\delta} > nx^2$, we have $\frac{\partial p_1}{\partial(t+1)} > 0$. Hence, the bound derived from (16) will monotonely decrease with increasing (t+1). When the upper bound of $(t+1)$ is available, the bound derived from (16) approaches its minimum.

According to (2), the value of x is approximately between -10db and 4db. When x is greater than 4db, the value of p_e^u is about zero. On the other hand, when x is less than -10db, the value of p_e^u is about 0.5. Considering (7), the value of x is on the interval [-10db, 4db]. Corresponding channel BER, p_e^u, is shown in Fig. 1.

In this paper, we adopt two common bounds: Plotkin bound and Hamming bound. That is, we assume that the linear block codes under research have maximum ability of error correction. Consequently, the bounds derived in next two subsections are the lower bounds for the linear block codes, which may be out of reach for many block codes in existence.

2.2 Coding Ratio under Plotkin Bound

According to communication theory, the maximum of minimum distance of linear block codes could be well approximated by Plotkin Bound in case of low coding ratio [14]. According to q-symbol Plotkin Bound, we have [14]

$$\begin{aligned} 2t+1 &\le \tfrac{nq^{k-1}(q-1)}{q^k-1} \\ t+1 &\le \tfrac{(n+1)q^k-nq^{k-1}-1}{2(q^k-1)} \end{aligned} \tag{18}$$

For large k, satisfying $q^{-k} \to 0$, above inequality reduces to

$$t+1 \le \frac{n(q-1)+q}{2q} \tag{19}$$

In case of $q=2$, the block code is binary, and (19) can be rewritten as:

$$t+1 \le \frac{n+2}{4} \tag{20}$$

Since the bound derived from (16) will monotonely decrease with increasing $(t+1)$, it is appropriate to assume that the block code under consideration has best correcting capability. Considering (12), (16) and (20), we have

$$\begin{aligned} \delta &= n^2x^4 + \tfrac{n(n+2)(3n-2)}{4}x^2r \\ &\ge \left(\tfrac{1}{x}Q^{-1}(\tfrac{[2(t+1)n+nx^2]-\sqrt{\delta}}{2(n^2+nx^2)})\right)^2 \end{aligned} \tag{21}$$

Fig. 2 shows the lower bound for binary block codes coding ratio, in case of n=*63, 127, 255, 511*. Obviously, the bound decreases with the increase of n. Note that the coding ratio can be very low when the x is large enough.

2.3 Coding Ratio under Hamming Bound

The maximum of minimum distance of linear block codes can be well approximated by Hamming Bound in case of high coding ratio [14]. From q-symbol Hamming Bound, we have [14]:

$$q^{n-k} \ge \sum_{i=0}^{t} \binom{n}{i} (q-1)^i \tag{22}$$

Considering $\sum\limits_{i=0}^{n} \binom{n}{i} (q-1)^i = q^n$, above inequality can be rewritten as

$$q^{n-k} \ge q^n \sum_{i=0}^{t} \binom{n}{i} (\frac{q-1}{q})^i (\frac{1}{q})^{n-i} \tag{23}$$

$$q^{-k} \ge \sum_{i=0}^{t} \binom{n}{i} (\frac{q-1}{q})^i (\frac{1}{q})^{n-i} \tag{24}$$

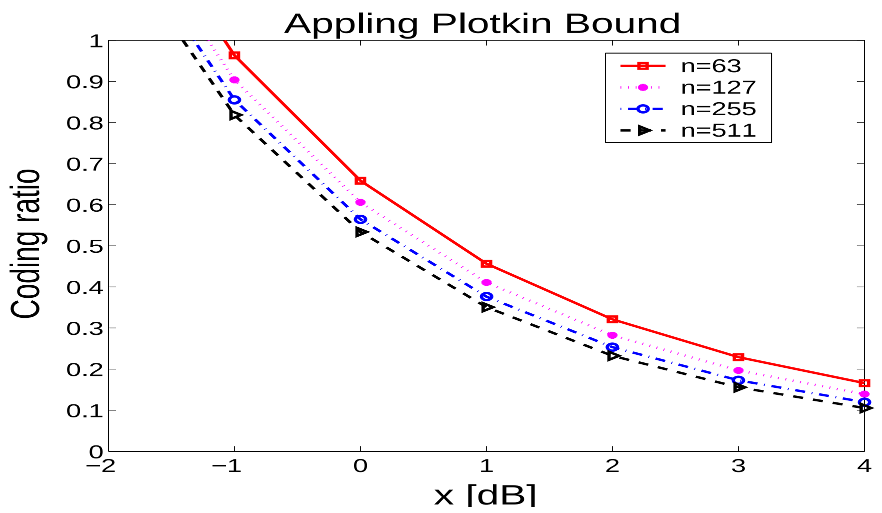


Fig. 2. Coding ratio estimation applying Plotkin Bound

The right side of above inequality can be viewed as a binomial distribution variable. Here again, we use Gaussian variable to approximate binomial distribution variable for moderate values of n. Then, we have

$$q^{-k} \geq \Phi(\frac{t - n(q-1)/q}{\sqrt{n(q-1)/q^2}}) \tag{25}$$

$$t + 1 \leq \frac{\sqrt{n(q-1)}}{q}\Phi^{-1}(q^{-k}) + \frac{n(q-1)}{q} + 1 \tag{26}$$

where $\Phi^{-1}(\bullet)$ is the inverse function of $\Phi(\bullet)$.

For large k, satisfying $q^{-k} \to 0$, and taking into account of (24) and the property of $\Phi(\bullet)$ (that is, $\Phi(4) = 1$, $\Phi(-4) = 0$), we have

$$\frac{t - n(q-1)/q}{\sqrt{n(q-1)/q^2}} \leq -4 \tag{27}$$

$$t + 1 \leq \frac{n(q-1) - 4\sqrt{n(q-1)}}{q} + 1 \tag{28}$$

In case of $q = 2$, the block code is binary, and (28) can be rewritten as

$$t + 1 \leq \frac{n}{2} - 2\sqrt{n} + 1 \tag{29}$$

Since the bound derived from (16) will monotonely decrease with increasing $(t + 1)$, it is appropriate to assume that the block code under consideration has

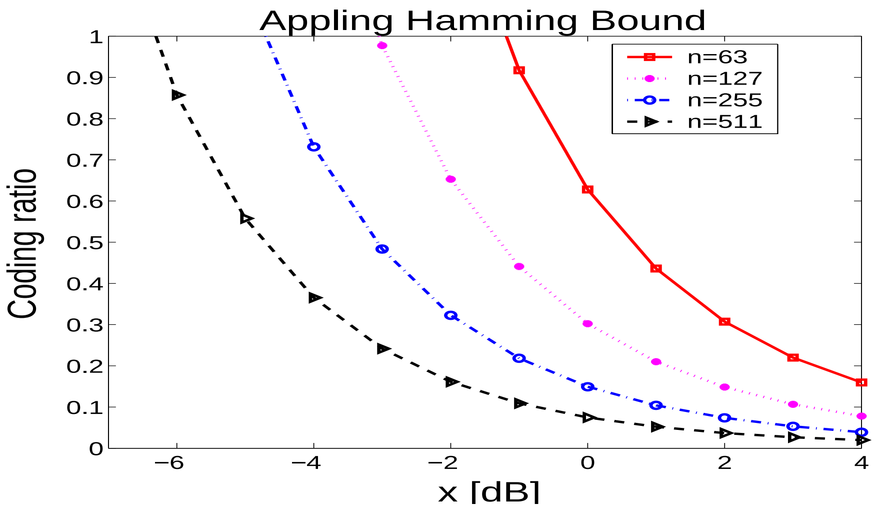


Fig. 3. Coding ratio estimation applying Hamming Bound

best correcting capability. Considering (12), (16) and (29), we have

$$\begin{aligned}\delta &= n^2x^4 + n[n^2 - (4\sqrt{n} - 2)^2]x^2 \\ r &\geq \left(\frac{1}{x}Q^{-1}\left(\frac{[2(t+1)n+nx^2]-\sqrt{\delta}}{2(n^2+nx^2)}\right)\right)^2\end{aligned} \tag{30}$$

Fig. 3 shows the lower bound for binary block codes coding ratio, in case of *n=63, 127, 255, 511*. Obviously, the bound decreases with the increase of n. Note that the coding ratio can be very low when the x is big enough, too.

3 Simulation Results

To verify the lower bound of block codes coding ratio, we compared the symbol error rates of two schemes. We adopt 64-bit binary sequence as the watermark and the 256*256*8 "Lena" image as the host image. Equation (1) is used as the embedding model. By adjusting the embedding strength, the two schemes had approximate PSNR (about 44 dB). That is, we decrease the embedding strength, S, for a long to-be-embedded sequence, and increase the embedding strength for a short to-be-embedded sequence. The two schemes are as follows.

(1) Block coded scheme. We chose mid-frequency coefficients of 8*8 DCT for watermarking. The block codes employed are binary and non-binary. In detection, we used "hard-decision" to get each bit.

(2) Uncoded scheme. The embedding and detection are similar to the former scheme. The only difference between this scheme and the former one is that there is no encoder and decoder of ECC. Hence, this scheme has a bigger embedding strength.

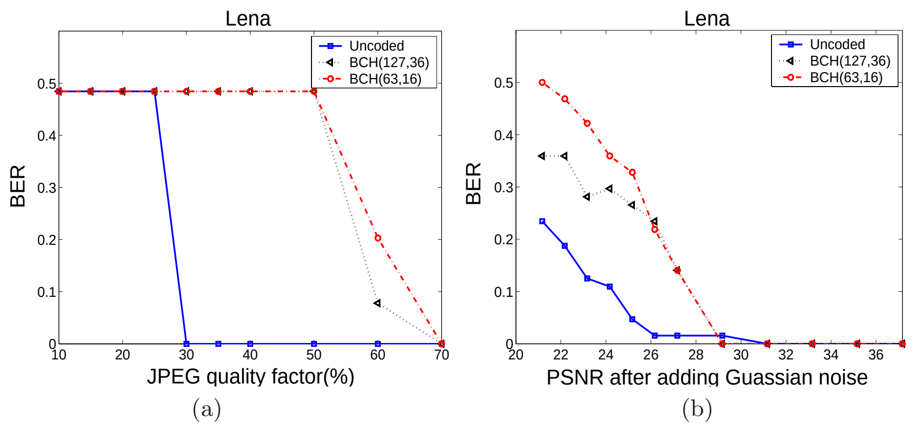


Fig. 4. (a) Performance under JPEG compression, (b) Performance under Gaussian noise attack

To verify our analysis, we choose JPEG compression and Gaussian noise corruption as attacks, since the two attacks can be approximated by AWGN to some extent.

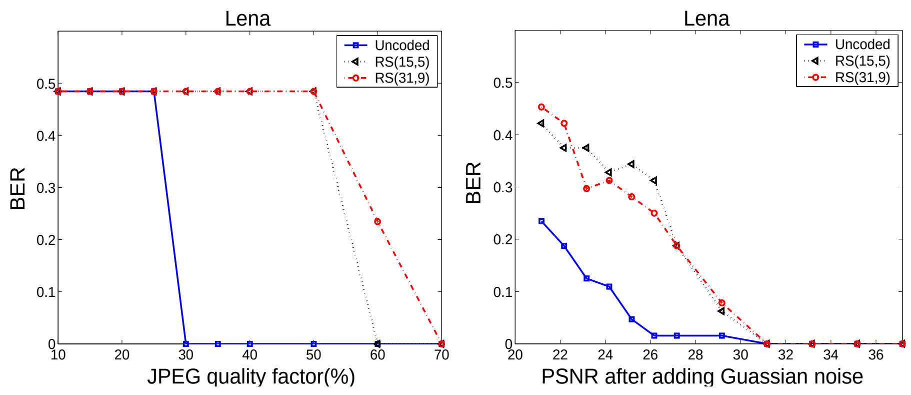
3.1 The Binary Code Case

Many binary block codes were used in watermarking systems to correct random error bits, including BCH codes, repetition codes, and their concatenation. In this paper, we choose BCH (127,36) and BCH (63, 16) to test the bounds derived in Section 2. Note that, coding ratios of the two codes are both about 0.25, and both below the bounds derived from Plotkin Bound and Hamming Bound on the interval [-3, 4] db. That is, the coding ratios in practice do not satisfy (21) or (30), and consequently the symbol error rate after decoding p^c_{sig} cannot satisfy (7). Recall the fact that ECC could bring no coding gain for information hiding whenever Inequality (7) is not satisfied. So, the two BCH codes here should have no coding gain.

Fig.4 shows that the uncoded scheme has lower BER under both JPEG compression and Gaussian noise corruption, comparing with schemes using BCH codes. So, the two BCH codes bring no coding gain. Thus, the experimental results support our previous analysis in this subsection.

3.2 The Non-Binary Code Case

Among various types of non-binary codes, Reed-Solomon (RS) codes are important for practical applications, and are often found in watermarking systems. In this paper, we chose RS (15, 5) and RS (31,9) to test the bounds derived in Section 2, with q=4, and 5 respectively. Since RS codes have the same coding ratio as their corresponding binary codes and the coding length of these two



(a) (b)

Fig. 5. (a) performance under JPEG compression, (b) Performance under Gaussian noise attack

RS codes are 60 and 153 bits respectively, it is reasonable to apply the coding ratio bounds for binary codes. Note that, coding ratios of this two codes are both below the bounds derived from Plotkin Bound and Hamming Bound on the interval [-3, 4] db. That is, the coding ratios in practice also do not satisfy (21) or (30), and consequently the symbol error rate after decoding p^c_{sig} cannot satisfy (7). Recall the fact that ECC could bring no coding gain for information hiding whenever Inequality (7) is not satisfied. So, the two RS codes here should have no coding gain.

Fig.5 shows that the uncoded scheme has lower BER under both JPEG compression and Gaussian noise corruption, comparing with schemes using RS codes. So, the two RS codes bring no coding gain. Thus, the experimental results support our previous analysis in this subsection.

4 Conclusions

In this paper, we have addressed an interesting issue, i.e., whether applying ECC can always lead to the robustness enhancement of watermark signal. The main contributions and conclusions are listed below.

Having emphasized the differences between watermark channel and common communication channel. That is, the imperceptibility of original media signal is a peculiar constraint in information hiding, which does not exist in common communication systems. In the latter cases, coding and modulation are independent. That is, the coding ratio has no effect on the amplitude of signal. On the contrary, the coding ratio greatly affects the amplitude of hidden signal in watermarking.

Based on this viewpoint, we analyzed the error rate of block codes, and present the bounds for coding ratio of binary and non-binary linear block codes in the application of watermarking and information hiding.

Experiments are made to test the bounds derived in this paper, and the experimental results support our analysis.

Though the bounds are obtained in the case of linear block codes, they can be easily extended to convolutional codes.

Acknowledgement

Authors appreciate the support received from NSFC (60325208, 60172067, 60133020), "863" Program (2002AA144060), NSF of Guangdong (013164), Funding of China National Education Ministry.

References

[1] F. A. P. Peticolas, R. J. Anderson and M. G. Kuhn, “Information hiding—a survey,” *Proceedings of the IEEE*, 1999,87(7): 1062-1078.

[2] Ingemar J. Cox, Matthew L. Miller, and Jeffery A. Bloom, “Digital Watermarking”, Academic Press,1999

[3] J. Huang and Y. Q. Shi, “Reliable Information Bit Hiding,” *IEEE Trans. on Circuits and Systems for Video Technology*, 2002, 12(10).

[4] F.Pérez-González, J. R. Hernández, and B. Felix, “Approaching the capacity limit in image watermarking: a perspective on coding techniques for data hiding applications,” Signal Processing, 2001, 81(6): 1215-1238.

[5] C. –F. Wu and W. –S. Hsieh, “Image refining technique using watermarking,” *IEEE Trans. on Consumer Electronics*, 2000, 46(1): 1-5.

[6] S. Pereira, S. Voloshynovskiy, and T. Pun, “Effective channel coding for DCT watermarking,” *Proc. IEEE Int. Conf. on Image Processing*, 2000, vol. 3, pp. 671-673.

[7] J. Huang, G. F. Elmasry, and Yun Q. Shi, “Power constrained multiple signaling in digital image watermarking,” *Proc. of IEEE Int. Workshop on Multimedia Signal Processing*, 1998, pp. 388-393.

[8] S. Baudry, J. F. Delaigle, B. Sankur, B. Macq, H. Maitre, “Analyses of Error Correction Strategies for Typical Communication Channels in Watermarking,” *Signal Processing,* 2001, 81(6): 1239-1250.

[9] F. Balado, F. P. Gonzalez, S. Scalise, “Turbo coding for sample-level watermarking in the DCT domain,” *Proc. of IEEE Int. Conf. on Image Processing*, 2001, Vol. 3, pp. 1003-1006.

[10] C. Desset, B. Macq, L. Vandendorpe, “Block error-correcting codes for systems with a very high BER: Theoretical analysis and application to the protection of watermarks”, *Signal Processing: Image Communication*, 2002,17(5): 409-421.

[11] M. -J. Tsai, K. –Y. Yu, and Y. –Z. Chen, “Joint Wavelet and Spatial Transformation for Digital Watermarking,” *IEEE Trans. on Consumer Electronics*, 2000, 46(1): 241-245.

[12] Zhigang Cao and Yasheng Qian, The Principle of Modern Communications, Tsinghua University Press, 1992.

[13] Limin Gu, Jiwu Huang, Yun Q. Shi, “Analysis of the Role Played by Error Correcting Coding in Robust Watermarking”, in: *Proc. of IEEE Int. Sym. on Circuits and Systems*, vol. 3, pp. 798-801, Bangkok, Thailand, 2003.

[14] Xinmei Wang and Guozhen Xiao, Error Correcting Codes: Principles and Methods, Xidian University Press, 1991.

Security Analysis of Some Proxy Signatures

Guilin Wang, Feng Bao, Jianying Zhou, and Robert H. Deng

Infocomm Security Department (ICSD)
Institute for Infocomm Research (I^2R)
21 Heng Mui Keng Terrace, Singapore 119613
http://www.i2r.a-star.edu.sg/icsd/
{glwang,baofeng,jyzhou,deng}@i2r.a-star.edu.sg

Abstract. A proxy signature scheme allows an entity to delegate his/her signing capability to another entity in such a way that the latter can sign messages on behalf of the former. Such schemes have been suggested for use in a number of applications, particularly in distributed computing where delegation of rights is quite common. Followed by the first schemes introduced by Mambo, Usuda and Okamoto in 1996, a number of new schemes and improvements have been proposed. In this paper, we present a security analysis of four such schemes newly proposed in [14, 15]. By successfully identifying several interesting forgery attacks, we show that these four schemes all are *insecure*. Consequently, the fully distributed proxy scheme in [11] is also insecure since it is based on the (insecure) LKK scheme [13, 14]. In addition, we point out the reasons why the security proofs provided in [14] are invalid.

Keywords. Digital signatures, proxy signatures, security analysis.

1 Introduction

Background. A proxy signature scheme allows one user Alice, called *original signer*, to delegate her signing capability to another user Bob, called *proxy signer*. After that, the proxy signer Bob can sign messages on behalf of the original signer Alice. Upon receiving a proxy signature on some message, a verifier can validate its correctness by the given verification procedure, and then is convinced of the original signer's agreement on the signed message. In other words, proxy signatures can be distinguished from standard signatures signed by either the original signer or the proxy signer. Proxy signature schemes have been suggested for use in a number of applications, particularly in distributed computing where delegation of rights is quite common. Examples include e-cash systems [21], mobile agents for electronic commerce [13, 14], mobile communications [22], grid computing [8], global distribution networks [3], and distributed shared object systems [17].

The basic idea of most existing proxy signature schemes is as follows. The original signer Alice sends a specific message with its signature to the proxy signer Bob, who then uses this information to construct a proxy private key. With the proxy private key, Bob can generate proxy signatures by employing a specified standard signature scheme. When a proxy signature is given, a verifier first computes the proxy public key from some public information, and then checks its validity according to the corresponding standard signature verification procedure.

Classification. Mambo, Usuda, and Okamoto introduced the concept of proxy signatures and proposed several constructions in [18, 19]. Based on the delegation type, they classified proxy signatures as *full delegation*, *partial delegation*, and *delegation by warrant*. In full delegation, Alice's private key is given to Bob so Bob has the same signing capability as Alice. For most of real-world settings, such schemes are obviously impractical and insecure. In a partial delegation scheme, a proxy signer has a new key, called *proxy private key*, which is different from Alice's private key. So, proxy signatures generated by using proxy private key are different from Alice's standard signatures. However, the proxy signer is not limited on the range of messages he can sign. This weakness is eliminated in delegation by warrant schemes by adding a warrant that specifies what kinds of messages are delegated, and may contain other information, such as the identities of Alice and Bob, the delegation period, etc.

According to another criterion, i.e., whether the original signer knows the proxy private key, proxy signatures can also be classified as *proxy-unprotected* and *proxy-protected* schemes. This differentiation is important in practical applications, since it enables proxy signature schemes to avoid potential disputes between the original signer and proxy signer. That is, in a proxy-protected scheme the original signer cannot first generate a proxy signature, and later claims that the signature is generated by the proxy signer, not by herself. Similarly, the original signer also cannot be misattributed by the proxy signer. Since they clearly distinguish the rights and responsibilities between the original signer and the proxy signer, the proxy-protected partial delegation by warrant schemes have attracted much more investigations than others. In fact, for simplicity, this special kind of schemes is often called as proxy signature scheme.

Security Requirements. The security requirements for proxy signature are first specified in [18, 19], and later are kept almost the same beside being enhanced by [13, 14]. That is, a secure proxy signature scheme should satisfy the following five requirements:

- R1) *Verifiability*: From the proxy signature, a verifier can be convinced of the original signer's agreement on the signed message.
- R2) *Strong unforgeability*: Only the designated proxy signer can create a valid proxy signature on behalf of the original signer. In other words, the original signer and other third parties who are not designated as proxy signers cannot create a valid proxy signature.
- R3) *Strong identifiability*: Anyone can determine the identity of the corresponding proxy signer from a proxy signature.

– R4) *Strong undeniability*: Once a proxy signer creates a valid proxy signature on behalf of an original signer, he cannot repudiate the signature creation against anyone else.
– R5) *Prevention of misuse*: The proxy signer cannot use the proxy key for purposes other than generating a valid proxy signature. In case of misuse, the responsibility of the proxy signer should be determined explicitly.

Related Work. Followed by the first constructions given in [18, 19], a number of new schemes and improvements have been proposed [12, 31, 32, 16, 9, 21, 22, 13, 14, 26, 15, 11, 30, 5]; however, most of them do not fully meet the above listed security requirements. In [12], Kim, Park and Won introduced the concept of partial delegation by warrant, and proposed a threshold proxy signature, in which the original singer's signing power is shared among a delegated group of n proxy singers such that only t or more of them can generate proxy signatures cooperatively. In [16], Lee et al. pointed out some weaknesses in Zhang's threshold proxy signatures [31, 32]. Later, some different opinions on their attacks are commented in [9]. In [13], Lee, Kim and Kim proposed non-designated proxy signature in which a warrant does not designate the identity of a proxy signer so any possible proxy signer can respond this delegation and become a proxy signer. Furthermore, their scheme is used to design secure mobile agents in electronic commerce setting [14]. One-time proxy signatures are studied in [2, 30].

In [15], Lee, Cheon, and Kim investigated whether a secure channel for delivery of a signed warrant is necessary in existing schemes. Their results show that if the secure channel is not provided, the MUO scheme [18] and the LKK scheme [13, 14] all are insecure. To remove the requirement of a secure channel and overcome some other weaknesses, they revised the MUO and LKK schemes. However, we will show that their revised schemes are still insecure. In addition, the PH scheme [23], which uses a 3-pass blind signature protocol to deliver the proxy private key, is also insecure [13, 15]. Boldyreva, Palacio, and Warinschi presented the formal definition and security notion for proxy signature in [5], i.e., the existential unforgeablity against adaptive chosen-message attacks [10]. At the same time, they proposed a provably secure scheme, called triple Schnorr proxy signature scheme, which is modified from the KPW scheme [12].

In contrast to the above mentioned schemes, which all are based on discrete logarithm cryptosystems, several RSA-based proxy signature schemes are proposed in [21, 14, 26]. In particular, the OTO scheme [21] and the LKK RSA-based scheme [14] are proved as secure as RSA signatures in the sense of polynomial-time reducibility. However, the OTO scheme in fact has a weak security since it is designed as a proxy-unprotected scheme; the security of Shao's schemes [26] is not proved. Moreover, the security proofs for the LKK schemes are incorrect (we will show later).

Our Contribution. In this paper, we present a security analysis of four proxy signature schemes newly proposed in [14, 15]. More specifically, we focus on the revised MUO scheme proposed by [15], the LKK scheme [13, 14] and its revised version [15], and the LKK RSA-based proxy signature proposed in [14]. To show the insecurity of these schemes, we successfully identify several forgery attacks which have different strengths and can be used in different settings. Moreover, we point out the reasons why the security proofs provided in [14] are invalid. In addition, the fully distributed proxy signature scheme in [11] is consequently broken, because it is based on the insecure LKK scheme.

Organization. In Section 2, the revised MUO scheme proposed by [15] is analyzed. Then, we analyze the security of the LKK schemes [14] and the revised LKK scheme [15] in Section 3. The security analysis of the LKK RSA-based scheme [14] is given in Section 4. Finally, we conclude our paper in Section 5.

2 The MUO Scheme and Its Derivatives

Throughout this paper, p, q are two large primes such that $q|(p-1)$ and $G_q = \langle g \rangle$ is a q-order multiplicative subgroup of $\mathbb{Z}_p^*$ generated by an element $g \in \mathbb{Z}_p^*$. The discrete logarithm problem in G_q is assumed to be difficult. Hereafter, we call three such integers (p, q, g) a DLP-triple. Let $h(\cdot)$ be a collision resistant hash function. In addition, it is assumed that Alice is the original signer with a *certified key pair* (x_A, y_A) where $y_A = g^{x_A} \bmod p$, and Bob is a proxy signer with a certified key pair (x_B, y_B) where $y_B = g^{x_B} \bmod p$. Here, a certified key pair (x_A, y_A) means that Alice knows the private key x_A and has to prove her knowledge of x_A when she registers her public key certificate with a CA. In fact, this is the recommended practice for certification authorities [1, 20], and can be used to prevent rogue-key attacks [5]. We denote by m_w the *warrant* which specifies the delegation period, what kind of message m is delegated, and the identity information of the original signer (and the proxy signer), etc.

2.1 The MUO Scheme and Its Security

Proxy Key Generation: Bob gets the proxy key pair (x_P, y_P) as follows [18].

1) The original signer Alice generates a random number $k \in \mathbb{Z}_q^*$ and computes $K = g^k \bmod p$. Then, she calculates $s_A = x_A + k \cdot K \bmod q$, and sends (s_A, K) to Bob through a secure channel.
2) Bob checks whether $g^{s_A} \equiv y_A \cdot K^K \bmod p$. If it is, Bob computes his proxy private key x_P as $x_P = s_A + x_B \cdot y_B \bmod q$.

Proxy Signature Generation: When the proxy singer Bob signs a document m on behalf of the original signer Alice, he executes one DLP-based ordinary signing algorithm [6, 7, 27] with the proxy private key x_P. The resulting proxy signature σ is denoted by $\sigma = (Sign(m, x_P), K, y_A, y_B)$.

Proxy Signature Verification: To verify the proxy signature σ, a verifier first computes the proxy public key y_P by

$$y_P = y_A \cdot K^K \cdot y_B^{y_B} \bmod p \; (= g^{x_P} \bmod p).$$

The verification is then carried out by the same checking operations as in the corresponding standard DLP signature scheme.

In the above MUO scheme, requirement R5 is not satisfied since the warrant (s_A, K) includes neither the identity information of the proxy signer nor the limit on delegated messages. Furthermore, the secure channel for delivery the warrant is necessary. Otherwise, an interesting attack demonstrated in [15] can be mounted if Schnorr signature scheme [27] is employed to generate proxy signatures. This attack enables an attacker to convert a proxy signature σ into a new one $\bar{\sigma}$, where $\bar{\sigma}$ is also a valid proxy signature but on behalf of a different original signer. We call this attack *Original Signer Changing Attack*.

2.2 Revised MUO Scheme and Its Security

To remove the secure channel while foiling the original signer changing attack, Lee, Cheon, and Kim proposed the following revised MUO scheme [15]. However, their revised scheme is still insecure, and in fact more vulnerable than the original MUO scheme. Since proxy signature generation and verification only involve some standard operations, we focus on the proxy key generation protocol only.

Revised Proxy Key Generation: Bob gets a proxy key pair (x_P, y_P) through the following two steps.

1) An original signer Alice generates a random number $k \in \mathbb{Z}_q^*$ and computes $K = g^k \bmod p$. After that, she calculates $s_A = x_A + k \cdot y_B \bmod q$, and then sends (s_A, K) to the proxy signer Bob (through a public, i.e., insecure, channel).
2) Bob checks whether $g^{s_A} \equiv y_A \cdot K^{y_B} \bmod p$. If (s_A, K) pass this validation, Bob computes the proxy private key as $x_P = s_A + x_B \cdot y_A \bmod q$.

The proxy public key y_P, which is used in the proxy signature verification stage, is generated as follows:

$$y_P = y_A \cdot K^{y_B} \cdot y_B^{y_A} \bmod p \ (= g^{x_P} \bmod p). \tag{1}$$

Now we discuss the security of the revised MUO scheme. We present three attacks to show that the revised MUO scheme *does not* satisfy R2, i.e., strong unforgeablility (and then does not satisfy R4, i.e., strong undeniability). In the first attack, the original signer can forge a valid proxy signature under the name of a proxy singer; our second attack allows an attacker to mount *impersonating attack*, i.e., he can generate valid proxy signature on any message by directly forging a valid proxy key pair; and in the third attack, a third party could frame Alice and Bob that they generated a proxy signature on an arbitrary message, but in fact they did not. In all three attacks, we assume that $y_B^{-1} \bmod q$ exists, i.e., $\gcd(y_B, q) = 1$. This assumption is reasonable since it holds for the overwhelming many of DLP public keys.

Forgery by the Original Signer. Suppose that the original signer Alice wants to generate valid proxy signatures which look as if they are generated by a proxy signer Bob. For this purpose, Alice does the following:

(1) Compute $b = -y_A \cdot y_B^{-1} \bmod q$ that satisfies $y_A + b \cdot y_B = 0 \bmod q$.
(2) Choose a random number $c \in \mathbb{Z}_q^*$, and define $\bar{K} = y_B^b \cdot g^c \bmod p$.
(3) The forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ is given by:

$$\bar{x}_P = x_A + c \cdot y_B \bmod q, \quad \text{and} \quad \bar{y}_P = g^{\bar{x}_P} \bmod p. \tag{2}$$

The forged $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$ is a valid proxy key pair since it satisfies equation (1), i.e.,

$$\begin{aligned} y_A \cdot \bar{K}^{y_B} \cdot y_B^{y_A} \bmod p &= y_A \cdot (y_B^b \cdot g^c)^{y_B} \cdot y_B^{y_A} \bmod p \\ &= g^{x_A + c \cdot y_B} \cdot y_B^{y_A + b \cdot y_B} \bmod p \\ &= g^{\bar{x}_P} \bmod p \\ &= \bar{y}_P. \end{aligned}$$

Using the above forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$, Alice can generate valid proxy signature on any message m.

Impersonating Attack. In this scenario, we assume that Bob is not designated as a proxy signer by the original signer Alice. However, our following attack enables Bob to become a proxy signer of Alice by forging a valid proxy key pair $(\bar{x}_P, \bar{y}_P)$, without Alice's agreement and authorization. For this sake, he operates as follows.

(1) Compute $a = -y_B^{-1} \bmod q$ that satisfies $a \cdot y_B + 1 = 0 \bmod q$.
(2) Choose a random number $c \in \mathbb{Z}_q^*$, and define $\bar{K} = y_A^a \cdot g^c \bmod p$.
(3) The forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ is given by

$$\bar{x}_P = c \cdot y_B + x_B \cdot y_A \bmod q, \quad \text{and} \quad \bar{y}_P = g^{\bar{x}_P} \bmod p. \tag{3}$$

Similarly, the validity of the forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$ can be verified, i.e., $\bar{y}_P = g^{\bar{x}_P} \bmod p = y_A \cdot \bar{K}^{y_B} \cdot y_B^{y_A} \bmod p$. Using this forged but valid $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$, Bob can generate valid proxy signatures for arbitrary messages of his choice. In addition, by defining $\bar{s}_A = c \cdot y_B \bmod q$, we have $g^{\bar{s}_A} = y_A \cdot \bar{K}^{y_B} \bmod p$. Therefore, different from the previous attack, in this attack Bob can provide $(\bar{s}_A, \bar{K})$ as a proof to show that he "indeed" is a proxy signer designated by the original signer Alice. It is easy to see that the distributions of $(\bar{s}_A, \bar{K}, \bar{x}_P, \bar{y}_P)$ and (s_A, K, x_P, y_P) are statistically indistinguishable. Thus, a third party cannot tell whether $(\bar{s}_A, \bar{K}, \bar{x}_P, \bar{y}_P)$ is a tuple generated normally. This attack implies that in the revised MUO scheme, any user, if he likes, "automatically" becomes a proxy singer of another user.

Framing Attack. In this attack, a third party Charlie can forge a proxy private key $\bar{x}_P$ and then generate valid proxy signatures such that a verifier believes that these signatures were signed by the proxy signer Bob on behalf of the original signer Alice. To accomplish the attack, Charlie does the followings.

(1) Compute $a = -y_B^{-1} \bmod q$, and $b = -y_A \cdot y_B^{-1} \bmod q$.
(2) Select a random number $c \in \mathbb{Z}_q^*$, and define $\bar{K} = y_A^a \cdot y_B^b \cdot g^c \bmod p$.
(3) The forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ is computed from

$$\bar{x}_P = c \cdot y_B \bmod q, \quad \text{and} \quad \bar{y}_P = g^{\bar{x}_P} \bmod p. \tag{4}$$

Again, it can be verified directly that the forged $(\bar{x}_P, \bar{y}_P)$ with $\bar{K}$ is a valid proxy key pair, i.e., equation (1) holds.

After getting the above forged but valid proxy key pair $(\bar{x}_P, \bar{y}_P)$, Charlie can select an arbitrary message m which is beneficial to himself. Then, by using the proxy private key $\bar{x}_P$ and $\bar{K}$, he generates a proxy signature $\bar{\sigma} = (Sign(m, \bar{x}_P), \bar{K}, y_A, y_B)$. Since $(\bar{x}_P, \bar{y}_P)$ and $\bar{K}$ satisfy the proxy public key verification equation (1), this forged proxy signature will pass the standard DLP-based signature verification equation respect to the proxy public key $\bar{y}_P$. When such a forged proxy signature is presented, Alice and Bob cannot deny that $\bar{\sigma}$ is Bob's proxy signature on behalf of Alice. Therefore, the result is that Alice and Bob will be framed. And the worst is that they cannot defend for themselves, because a judge cannot tell whether they are framed or they collude together to cheat the verifier.

In addition, we would like to point out that the revised MUO scheme has other two weaknesses. The first one is that if Bob got a warrant (s_A, K) from Alice, he can create a new one $(\bar{s}_A, \bar{K})$ for himself by selecting a random number $\bar{k} \in_R \mathbb{Z}_q^*$, and defining $\bar{K} = K \cdot g^{\bar{k}} \bmod p$ and $\bar{s}_A = s_A + \bar{k} \cdot y_B \bmod q$. In addition, from Bob's warrant (s_A, K) (remember in essence (s_A, K) is public information), Charlie can derive a new warrant $(\bar{s}, \bar{K})$ for himself if q does not divide his public key y_C. That is, he first computes $l = y_B \cdot y_C^{-1} \bmod q$, and selects a random number $\bar{k} \in \mathbb{Z}_q^*$. Then, Charlie defines $\bar{s}_A = s_A + \bar{k} \cdot y_C \bmod q$ $(= x_A + (kl + \bar{k}) \cdot y_C \bmod q)$, and $\bar{K} = K^l \cdot g^{\bar{k}} \bmod p$. Since $g^{\bar{s}_A} = y_A \cdot \bar{K}^{y_C} \bmod p$, Charlie gets his proxy private key $\bar{x}_P$ by $\bar{x}_P = \bar{s}_A + x_C \cdot y_A \bmod q$.

3 The LKK Scheme and Its Derivatives

3.1 The LKK Scheme and Its Security

Proxy Key Generation: The original signer Alice uses the Schonrr scheme [27] to sign a warrant m_w, which specifies which messages Bob can sign on behalf of Alice, the validity period of delegation, etc. That is, Alice chooses a random number $k_A \in \mathbb{Z}_q^*$, computes $r_A = g^{k_A} \bmod p$ and $s_A = k_A + x_A \cdot h(m_w, r_A) \bmod q$. Then, the tuple (m_w, r_A, s_A) is sent to the proxy signer Bob, and Bob checks its validity by

$$g^{s_A} \equiv y_A^{h(m_w, r_A)} \cdot r_A \bmod p.$$

If this verification is correct, Bob sets his proxy key pair (x_P, y_P) as

$$x_P = s_A + x_B \bmod q, \quad y_P = g^{x_P} \bmod p \ (= y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B \bmod p).$$

Proxy Signature Generation: With the proxy key pair (x_P, y_P), Bob can use any DLP-based signature scheme to generate proxy signature on any delegated message m. The resulting proxy signature is a tuple $\sigma = (sign(m, x_P), m_w, r_A, y_A, y_B)$.

Proxy Signature Verification: To check the validity of σ, a verifier first checks whether message m conforms to the warrant m_w. If this check is positive, he computes the proxy public key y_P as follows:

$$y_P = y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B \bmod p \ (= g^{x_P} \bmod p). \tag{5}$$

Finally, the verifier checks whether $sign(m, x_P)$ is a valid signature on message m with respect to the proxy public key y_P in the corresponding DLP-based signature scheme.

The authors of [14] proved that the (Schonrr-based) LKK scheme is as secure as the Schnorr scheme. Then, based on the result of [24] which shows that under DLP assumption the Schnorr scheme is secure in the random oracle model [4], they concluded that the LKK satisfies all the requirements from R1 to R5. However, the proofs provided in [14] is not rigorous. For example, when they discuss the forgery by Alice in Theorem 1 (page 480 of [14]), the ability of Alice is limited to forge a Schnorr signature pair (r, s) for a new message m such that $g^s = (y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B)^{h(m,r)} \cdot r \bmod p$, where Alice knows two exponents k_A and k such that $r_A = g^{k_A} \bmod p$ and $r = g^k \bmod p$. However, according to the verification procedure described in their scheme, a successful forgery by Alice only means that she can generate a Schnorr signature pair (r, s) for a new message m that satisfies $g^s = (y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot \bar{r}_A \cdot y_B)^{h(m,r)} \cdot r \bmod p$, where the warrant $\bar{m}_w$ may be different from m_w though m conforms to $\bar{m}_w$, and $\bar{r}_A$ and r are two public values but Alice maybe *does not* know their discrete logarithms to the base g (In the extreme situation, $\bar{r}_A, r \notin G_q$). In fact, the original signer Alice can mount the following concrete attack.

(1) Create a warrant $\bar{m}_w$, select a random number $c \in \mathbb{Z}_q^*$, and then define $\bar{r}_A = y_B^{-1} \cdot g^c \bmod p$.
(2) The forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ is given by:

$$\bar{x}_P = c + x_A \cdot h(\bar{m}_w, \bar{r}_A) \bmod q, \quad \bar{y}_P = g^{\bar{x}_P} \bmod p.$$

One can directly verify that $(\bar{x}_P, \bar{y}_P)$ is a valid proxy key pair with respect to $(\bar{m}_w, \bar{r}_A)$, since $\bar{y}_P = g^{\bar{x}_P} \bmod p = y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot \bar{r}_A \cdot y_B \bmod p$.

After getting $\bar{x}_P$, for any message m which conforms to $\bar{m}_w$, Alice can generate a valid proxy signature σ for m in which Bob will be identified as the proxy signer. Bob is thus framed that he signed some message on behalf of Alice. When such a forged proxy signature is presented, a verifier cannot recognize that in fact it was not generated by Bob. In the above attack, Alice cannot provide $\bar{s}_A$ satisfying $g^{\bar{s}_A} = y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot \bar{r}_A \bmod p$. So, this attack is more powerful if the original signer is not supposed to present s_A when a dispute between Alice and Bob happens. For example, in electronic commerce setting where mobile agents are used to transfer signed warrants [14], all or some s_A's may be deleted before potential disputes occur due to the storage consideration. Anyway, neither the original LKK scheme nor the revised LKK scheme presented how to solve disputes between the original signer and proxy signers. At the same time, the authors also did not mention whether there is a need to keep s_A and x_P safely after they have been used.

Lee et al. [15] found that in the Schnorr-based LKK scheme, the original signer can change Bob's standard Schnorr signatures into proxy signatures in which Bob is designated as the proxy signer, and vice versa. We call this attack *Transferring Attack*. Their attack is weaker than our above attack in two respects. On the one hand, our attack allows the original signer to forge proxy signature on *any* message as her will, but in the transferring attack she cannot forge proxy signatures for all messages which the proxy signer never signs. On the other hand, we note that the transferring attack can be eliminated by a simple technique - padding. For example, we can modify the LKK scheme as follows. When the proxy signer Bob wants to generate a proxy signature for a message m, he signs on the padded message $m||m_w$, instead of m. That is, here, the warrant m_w is viewed as a proxy signature identifier. At the same time, a proxy signer should not generate standard signatures on a message of the form $m||m_w$. With this modification, it is now difficult for the original signer Alice to apply transferring attacks.

Furthermore, to remove the secure channel and avoid the transferring attack of the original LKK scheme, Lee et al. proposed a revised LKK scheme [15]. In the next subsection, however, we will show that their revised LKK scheme is still insecure.

3.2 Revised LKK Scheme and Its Security

Lee et al.'s revised LKK scheme [15] is reviewed as follows.

Revised Proxy Key Generation: Bob gets a proxy key pair (x_P, y_P) from the original signer Alice as follows.

1) Alice chooses at random $k_A \in \mathbb{Z}_q^*$, and computes $r_A = g^{k_A} \bmod p$ and $s_A = k_A + x_A \cdot h(m_w, r_A) \bmod q$. Then, the tuple (m_w, r_A, s_A) is sent to Bob through a public channel.
2) Bob checks whether (r_A, s_A) is a valid Schnorr signature on the warrant m_w. If the answer is positive, Bob chooses a random number k_P, computes $r_P = g^{k_P} \bmod p$, and then sets his proxy key pair (x_P, y_P) as

$$x_P = s_A + r_P \cdot x_B \bmod q, \quad \text{and} \quad y_P = g^{x_P} \bmod p.$$

Proxy Signature Generation and Verification: Using the proxy private key x_P and public parameters (m_w, r_A, r_P), the proxy singer Bob can generate a standard

DLP-based signature for any message m which conforms to the warrant m_w. Let $\sigma = (sing(m, x_P), m_w, r_A, r_P)$ [1] be Bob's proxy signature on message m, a verifier checks the validity of σ with respect to the proxy public key y_P, which is computed by:

$$y_P = y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B^{r_P} \bmod p \; (= g^{x_P} \bmod p). \tag{6}$$

Lee et al. claimed that the above revised LKK scheme not only removes the requirement on the secure channel, but also overcomes the weaknesses in previous schemes. They indeed provide a security analysis to show that the revised LKK scheme is immune to the transferring attack. However, we find their revised version is still vulnerable to the forgery attack by the original signer and the original signer changing attack.

Forgery by the Original Singer. To forge a proxy key pair, the original signer Alice selects two random numbers $\bar{k}_P, c \in \mathbb{Z}_q^*$, sets $\bar{r}_P = g^{\bar{k}_P} \bmod p$ and defines $\bar{r}_A = y_B^{-\bar{r}_P} \cdot g^c \bmod p$. Then, Alice computes the forged proxy key pair $(\bar{x}_P, \bar{y}_P)$ as

$$\bar{x}_P = c + x_A \cdot h(\bar{m}_w, \bar{r}_A) \bmod q, \quad \bar{y}_P = g^{\bar{x}_P} \bmod p.$$

In the above equation, $\bar{m}_w$ is a new warrant created by Alice. In addition, the forged $(\bar{x}_P, \bar{y}_P)$ with $(\bar{m}_w, \bar{r}_A, \bar{r}_P)$ is a valid proxy key pair, since

$$\begin{aligned} y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot \bar{r}_A \cdot y_B^{\bar{r}_P} &= y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot y_B^{-\bar{r}_P} \cdot g^c \cdot y_B^{\bar{r}_P} \bmod p \\ &= y_A^{h(\bar{m}_w, \bar{r}_A)} \cdot g^c \bmod p \\ &= g^{c + x_A \cdot h(\bar{m}_w, \bar{r}_A)} \bmod p \\ &= g^{\bar{x}_P} \bmod p \\ &= \bar{y}_P. \end{aligned}$$

Using the forged key pair $(\bar{x}_P, \bar{y}_P)$, Alice can create valid proxy signatures for arbitrary messages at her will.

Original Signer Changing Attack. Now, assume that Bob has a proxy key pair (x_P, y_P) for the original signer Alice. At the same time, Bob generated a proxy signature $\sigma = (r, s, m_w, r_A, r_P)$ for a message m using the Schnorr signature scheme. That is, there exists a number $k \in \mathbb{Z}_q^*$ such that

$$r = g^k \bmod p, \quad s = k + x_P \cdot h(m, r) \bmod q.$$

In the following, we show that a third party Charlie can transform σ into $\bar{\sigma}$ such that $\bar{\sigma}$ is a proxy signature generated by Bob on behalf of Charlie himself. To this end, Charlie first creates a new warrant $\bar{m}_w$ such that the message m conforms to $\bar{m}_w$. In addition, if necessary, Charlie specifies in $\bar{m}_w$ that himself

[1] If y_A and y_B are not included in m_w, it is necessary to list them in σ explicitly.

is the original signer, and Bob is the proxy signer. After $\bar{m}_w$ has been prepared, Charlie chooses a random number $k_C \in \mathbb{Z}_q^*$, and computes $r_C = g^{k_C} \bmod p$ and $s_C = k_C + x_C \cdot h(\bar{m}_w, r_C) \bmod q$. Finally, using the public information s_A, Charlie computes $\bar{s}$ by:

$$\bar{s} = s - s_A \cdot h(m,r) + s_C \cdot h(m,r) \bmod q$$

The resulting proxy signature $\bar{\sigma}$ for message m is set as $\bar{\sigma} = (r, \bar{s}, \bar{m}_w, r_C, r_P)$.

To verify the validity of $\bar{\sigma}$, a recipient first generates the proxy public key $\bar{y}_P$ from

$$\bar{y}_P = y_C^{h(\bar{m}_w, r_C)} \cdot r_C \cdot y_B^{r_P} \bmod p.$$

Let $\bar{x}_P = r_P \cdot x_B + s_C \bmod q$, then we have $\bar{y}_P = g^{\bar{x}_P} \bmod p$. The following equalities justify that $\bar{\sigma}$ is a valid proxy signature generated by Bob on behalf of Charlie, i.e., the tuple $(r, \bar{s})$ is a valid Schnorr signature pair with respect to the key pair $(\bar{x}_P, \bar{y}_P)$:

$$\begin{aligned}
\bar{s} &= s - s_A \cdot h(m,r) + s_C \cdot h(m,r) \bmod q \\
&= k + x_P \cdot h(m,r) - s_A \cdot h(m,r) + s_C \cdot h(m,r) \bmod q \\
&= k + (r_P \cdot x_B + s_A) \cdot h(m,r) - s_A \cdot h(m,r) + s_C \cdot h(m,r) \bmod q \\
&= k + (r_P \cdot x_B + s_C) \cdot h(m,r) \bmod q \\
&= k + \bar{x}_P \cdot h(m,r) \bmod q.
\end{aligned}$$

Furthermore, it is not difficult to see that when Bob has two signed warrants (m_w, r_A, s_A) and $(\bar{m}_w, r_C, s_C)$ generated by Alice and Charlie respectively, anybody can convert Bob's proxy signature σ on a message m on behalf of the original signer Alice into a new proxy signature $\bar{\sigma}$ on the same message in which Bob is the proxy signer but Charlie becomes the original signer. The only requirement is that message m also conforms to the warrant $\bar{m}_w$. The reason is that (r_A, s_A) and (r_C, s_C) are in fact public information, since they are transferred through a public channel.

4 The LKK RSA-Based Scheme

In this section we analyze the security of the LKK RSA-based proxy scheme proposed in [14]. First of all, we review it briefly.

Let $((n_A, e_A), d_A)$ and $((n_B, e_B), d_B)$ be the certified ordinary RSA key pairs of Alice and Bob, respectively. Denote the identities of Alice and Bob as ID_A and ID_B. To designate Bob as her proxy signer, Alice creates a warrant m_w and signs it, i.e., she generates $k = h(ID_A, m_w)^{d_A} \bmod n_A$. Then, (ID_A, m_w, k) is sent to Bob.

Bob checks the validity of (ID_A, m_w, k) by $h(ID_A, m_w) \equiv k^{e_A} \bmod n_A$. To generate a proxy signature on message m which conforms to m_w, he computes the triple (x, y, z) as follows:

$$\begin{aligned}
x &= h(ID_A, m_w, ID_B, m)^{d_B} \bmod n_B, \\
y &= h(ID_A, m_w)^x \bmod n_A, \\
z &= k^x \bmod n_A.
\end{aligned}$$

The resulting proxy signature is $\sigma = (ID_A, m_w, ID_B, m, x, y, z)$.

A recipient accepts a proxy signature σ iff all the following checks hold:

1) Verify whether m conforms to the warrant m_w.
2) Verify Bob's RSA signature: $h(ID_A, m_w, ID_B, m) \equiv x^{e_B} \bmod n_B$.
3) Verify the validity of y: $y \equiv h(ID_A, m_w)^x \bmod n_A$.
4) Verify Alice's RSA signature: $y \equiv z^{e_A} \bmod n_A$.

In [14], Lee et al. proved that the above RSA-base proxy signature is as secure as the RSA signature scheme. Obviously, forgery by Alice is difficult since she has to forge Bob's standard RSA signature x on message (ID_A, m_w, ID_B, m). However, whether Bob can mount some attack is another story. Similar to previous discussion presented in Section 3.1, one can find that their proof is not rigorous. And in fact, the reasoning about Bob's attacking ability in the proof of Theorem 2 (page 482 of [14]) is incorrect, since given the value of x Bob cannot find the value of $1/x \bmod \phi(n_A)$. In more detail, the authors first concluded that if Bob can generate a valid proxy signature σ without Alice's delegation, then the following equations hold:

$$z = y^{d_A} \bmod n_A = h(ID_A, m_w)^{xd_A} \bmod n_A.$$

Secondly, the authors claimed that the above equalities mean that Bob can compute Alice's RSA signature k on message (ID_A, m_w) from

$$k = z^{1/x} \bmod n_A \ (= h(ID_A, m_w)^{d_A} \bmod n_A).$$

So the authors have assumed that Bob can get the value of $1/x \bmod \phi(n_A)$ when the value of x is given. However, this is the RSA problem [25] so it is an intractable problem for Bob! Therefore, Theorem 2 provided by [14] is invalid.

To show the insecurity of the RSA-based LKK scheme directly, we demonstrate the following attack for Bob who is not designated as a proxy signer of Alice.

(1) Try different warrants to get an m_w such that m conforms to m_w, and that $e_A | x$. The latter condition means that $x = h(ID_A, m_w, ID_B, m)^{d_B} \bmod n_B = e_A \cdot b$ for an integer b.
(2) Compute $y = h(ID_A, m_w)^x \bmod n_A$ and $z = h(ID_A, m_w)^b \bmod n_A$.
(3) Output the proxy signature $\sigma = (ID_A, m_w, ID_B, m, x, y, z)$.

It is not difficult to check the correctness of the above attack, since we have $y \equiv h(ID_A, m_w)^x \bmod n_A = h(ID_A, m_w)^{e_A \cdot b} \bmod n_A = z^{e_A} \bmod n_A$. The smaller e_A is, the easier is to mount our attack. Suggested values for e_A in practice include 3, 17, and $2^{16} + 1 = 65537$ (X.509 recommends 65537, and PKCS #1 recommends either 3 or 65537). If $e_A = 3$, our attack succeeds for one try of m_w with probability about 1/3. Even for $e_A = 65537$, one required warrant can be found after about 65537 different tries. This is not a hard task for an attacker.

The LKK RSA-based scheme has another weakness, that is, it is not very efficient. In both proxy signature generation and verification, one entity has to perform three RSA modular exponentiations. Using the generic construction given in [5], one can derive a provable secure RSA-based proxy signature scheme in which only two modular exponentiations are needed in both proxy signature generation and verification. The basic idea is very simple. That is, to delegate her signing ability to Bob, Alice sends Bob a proxy certificate `cert` which is her standard RSA-signature on the warrant m_w. A proxy signature σ on a message m consists of $(m, s, m_w, \texttt{cert})$, where s is Bob's standard RSA signature on message m. A verifier validates σ by performing the following three checks: (a) Whether m conforms to the warrant m_w; (b) Whether `cert` is Alice's RSA signature on the warrant m_w; (c) And whether s is Bob's RSA signature on the message m. To provide provable security, m_w and m may need to be padded in some way. For more detail, please refer to [5].

5 Conclusion

In this paper, we presented a security analysis of four proxy signature schemes newly proposed in [14, 15]. Our results show that all these schemes are insecure, i.e., forgeable. As a by-product, the fully distributed proxy signature scheme in [11] is also broken, because it is based on the insecure LKK scheme [14]. In addition, we pointed out that the security proofs provided in [14] are incorrect.

References

[1] C. Adams and S. Farrell. Internet X.509 public key infrastructure: Certificate management protocols. *RFC 2510*, March 1999. 308

[2] M. Ai-Ibrahim and A. Cerny. Proxy and threshold one-time signatures. In: *Applied Cryptography and Network Security* (*ACNS'03*), LNCS 2846, pp. 123-136. Springer-Verlag, 2003. 307

[3] A. Bakker, M. Steen, and A. S. Tanenbaum. A law-abiding peer-to-peer network for free-software distribution. In: *IEEE International Symposium on Network Computing and Applications* (*NCA'01*), pp. 60-67. IEEE, 2001. 305

[4] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In: *Proc. of 1st ACM Conference on Computer and Communications Security* (*CCS'93*), pp. 62-73. ACM Press, 1993. 312

[5] A. Boldyreva, A. Palacio, and B. Warinschi. Secure proxy signature schemes for delegation of signing rights. Available at `http://eprint.iacr.org/2003/096` 307, 308, 317

[6] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, July 1985, IT-31(4): 469-472. 308

[7] FIPS 186. *Digital Signature Standard*. U. S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA, 1994. 308

[8] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In: *Proc. 5th ACM Conference on Computer and Communications Security* (*CCS'98*), pp. 83-92. ACM Press, 1998. 305

[9] H. Ghodosi and J. Pieprzyk. Repudiation of cheating and non-repudiation of Zhang's proxy signature schemes. In: *Information Security and Privacy* (*ACISP'99*), LNCS 1587, pp. 129-134. Springer-Verlag, 1999. 307

[10] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, April 1988, 17(2): 281-308. 307

[11] J. Herranz and Sáez. Verifiable secret sharing for general access structures, with applications to fully distributed distributed proxy signatures. In: *Financial Cryptography* (*FC'03*), LNCS 2742, pp. 286-302. Springer-Verlag, 2003. 305, 307, 308, 317

[12] S. Kim, S. Park, and D. Won. Proxy signatures, revisited. In: *Information and Communications Security* (*ICICS'97*), LNCS 1334, pp. 223-232. Springer-Verlag, 1997. 307

[13] B. Lee, H. Kim, and K. Kim. Strong proxy signature and its applications. In: *Proceedings of the 2001 Symposium on Cryptography and Information Security* (*SCIS'01*), Vol. 2/2, pp. 603-608. Oiso, Japan, Jan. 23-26, 2001. 305, 306, 307

[14] B. Lee, H. Kim, and K. Kim. Secure mobile agent using strong non-designated proxy signature. In: *Information Security and Privacy* (*ACISP'01*), LNCS 2119, pp. 474-486. Springer-Verlag, 2001. 305, 306, 307, 308, 312, 313, 315, 316, 317

[15] J.-Y. Lee, J. H. Cheon, and S. Kim. An analysis of proxy signatures: Is a secure channel necessary? In: *Topics in Cryptology - CT-RSA 2003*, LNCS 2612, pp. 68-79. Springer-Verlag, 2003. 305, 307, 308, 309, 313, 317

[16] N.-Y. Lee, T. Hwang, and C.-H. Wang. On Zhang's nonrepudiable proxy signature schemes. In: *Information Security and Privacy* (*ACISP'98*), LNCS 1438, pp. 415-422. Springer-Verlag, 1998. 307

[17] J. Leiwo, C. Hanle, P. Homburg, and A. S. Tanenbaum. Disallowing unauthorized state changes of distributed shared objects. In: *Information Security for Global Information Infrastructures* (*SEC'00*), pp. 381-390. Kluwer, 2000. 305

[18] M. Mambo, K. Usuda, and E. Okamoto. Proxy signature: Delegation of the power to sign messages. *IEICE Trans. Fundamentals*, Sep. 1996, Vol. E79-A, No. 9, pp. 1338-1353. 306, 307, 308

[19] M. Mambo, K. Usuda, E. Okamoto. Proxy signatures for delegating signing operation. In: *Proc. of 3rd ACM Conference on Computer and Communications Security* (*CCS'96*), pp. 48-57. ACM Press, 1996. 306, 307

[20] M. Meyers, C. Adams, D. Solo, and D. Kemp. Internet X.509 certificate request format. RFC 2511, March 1999. 308

[21] T. Okamoto, M. Tada, and E. Okamoto. Extended proxy signatures for smart cards. In: *Information Security Workshop* (*ISW'99*), LNCS 1729, pp. 247-258. Springer-Verlag, 1999. 305, 307

[22] H.-U. Park and I.-Y. Lee. A digital nominative proxy signature scheme for mobile communications. In: *Information and Communications Security* (*ICICS'01*), LNCS 2229, pp. 451-455. Springer-Verlag, 2001. 305, 307

[23] H. Petersen and P. Horster. Self-certified keys - concepts and applications. In: *Proc. of 3rd Conference on Communications and Multimedia Security*, pp. 102-116. Chapman & Hall, 1997. 307

[24] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3): 361-369, 2000. 312

[25] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, Feb. 1978, 21(2): 120-126. 316

[26] Z. Shao. Proxy signature schemes based on factoring. *Information Processing Letters*, 2003, 85: 137-143. 307

[27] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptography*, 1991, 4(3): 161-174. 308, 309, 311

[28] H.-M. Sun. Threshold proxy signatures. *IEE Proc.-Computers & Digital Techniques*, Sept. 1999, 146(5): 259-263.

[29] H.-M. Sun and B.-T. Hsieh. On the security of some proxy signature schemes. `http://eprint.iacr.org/2003/068.`

[30] H. Wang and J. Pieprzyk. Efficient one-time proxy signatures. In: *ASIACRYPT 2003*, LNCS 2894, pp. 507-522. Springer-Verlag, 2003. 307

[31] K. Zhang. Threshold proxy signature schemes. In: *Information Security Workshop* (*ISW'97*), LNCS 1396, pp. 282-290. Springer-Verlag, 1997. 307

[32] K. Zhang. Nonrepudiable proxy signature schemes. Manuscript, 1997. Available at `http://citeseer.nj.nec.com/360090.html` 307

A More Secure and Efficacious TTS Signature Scheme

Jiun-Ming Chen[1] and Bo-Yin Yang[2]*

[1] Chinese Data Security, Inc. *and* Mathematics Department
National Taiwan University, Taipei, Taiwan
jmchen@math.ntu.edu.tw
[2] Mathematics Department, Tamkang University, Tamsui, Taiwan
by@moscito.org

Abstract. In 2002 the authors introduced the new genre of digital signature scheme TTS (Tame Transformation Signatures) along with a sample scheme TTS/2. TTS is from the family of multivariate cryptographic schemes to which the NESSIE primitive SFLASH also belongs. It is a realization of T. Moh's theory ([31]) for digital signatures, based on Tame Transformations or Tame Maps. Properties of multivariate cryptosystems are determined mainly by their central maps. TTS uses Tame Maps as their central portion for even greater speed than C^*-derived SFLASH family of schemes, which uses monomials in a large field for the central portion, previously usually acknowledged as fastest.
We show a small flaw in TTS/2 and present an improved TTS implementation which we call TTS/4. We will examine in some detail how well TTS/4 performs, how it stands up to previously known attacks, and why it represents an advance over TTS/2. Based on this topical assessment, we consider TTS in general and TTS/4 in particular to be competitive or superior in several aspects to other schemes, partly because the theoretical roots of TTS induce many good traits. One specific area in which TTS/4 should excel is in low-cost smartcards. It seems that the genre has great potential for practical deployment and deserves further attention by the cryptological community.

Keywords. Multivariate, public-key, digital signature, finite field, tame transformation.

1 Introduction

Trapdoor mappings are central to Public-Key Cryptography. As such, cryptographers have studied trapdoor permutations and maps since the dawn of public key cryptography ([13]). A handful of the many schemes attempted reached practical deployment. However, *the critical trapdoor maps are often very slow, and that is frequently due to the sheer size of the algebraic structure.* Typical are the modular exponentiation in RSA or the discrete logarithms in ElGamal/DSA/ECC.

* Corresponding author

Multivariate public-key cryptosystems were born partly to circumvent this limitation. Great names like Shamir ([36]) and Diffie ([17]) abound among the pioneers, but the first scheme to show promise and grab everyone's attention was C^* by Imai and Matsumoto ([24, 30]). Unfortunately a few years later C^* was broken by Patarin ([37]) who in turn introduced many of his own innovations, some (HFE/C^{*-} families) of which are still extant. The variations on this theme seem endless.

TTS (Tame Transformation Signatures) digital signature schemes belong to this extended family. We propose TTS/4, an improvement variant of TTS and discuss some design, performance, and security issues associated with the TTS genre. Multivariates mainly differ in their central maps, or *kernels*, that determine the trapdoors and hence their security. We aim to show that the Tame Transformation, a *biregular map*[1] first introduced by T. Moh to cryptography, is a viable kernel for trapdoor permutations with appreciable gains versus other schemes, and one that warrants further investigation.

Sec. 2 is a succinct summary of TTS with theory and example. Like other multivariate schemes, TTS can be flexible in terms of hash length and is easily adaptable to 256-bit or longer hashes if needed, but TTS/4 is designed to work with current 160-bit hashes like SHA-1. We will quantify how well TTS/4 does by various metrics of speed and key size in Sec. 3. It compares well with some better-known alternatives. We see that TTS/4 is especially fast in signing and should be very suitable for use on a smartcard as seen in a point-by-point comparison with the SFLASH^{v2} scheme recommended by the NESSIE (New European Schemes for Signatures, Integrity, and Encryption) project for the same purpose ([1, 35]).

Avoiding the pitfalls that ensnared other schemes should be central to design decisions taken in present and future schemes, hence the multiplitude of techniques of Sec. 4 serves as an illustrative backdrop to TTS/4. We show that the TTS family can be made safe against current attacks.

2 Tame Transformation and TTS

While clock speeds went up according to Moore's law, unfortunately so did the complexity skyrocket and key lengths exponentiate. In a quarter-century, no alternative to the venerable RSA ever showed enough of a speed gain to become the heir apparent. Indeed, multivariate public-key cryptography arose out of this need for faster algorithms.

Cryptographers also applied their considerable talents to seeking faster alternatives in *birational* permutations ([17, 43]) over two decades. Birational implies *being polynomial or rational, with a polynomial or rational inverse*. Regrettably, an explicit low-degree inverse that brings swiftness in execution sometimes has the undesirable side effect of engendering vulnerabilities ([7, 8, 17]).

T. Moh first took *Tame Transformations* into the landscape of Cryptography from their native habitat of Algebraic Geometry ([31]). A Tame Transformation

[1] A bijective map that is polynomial both ways.

over a field K, which shall be $\mathrm{GF}(2^8)$ unless otherwise specified, is either invertible and affine ($\mathbf{x} \mapsto \mathbf{y} = \mathsf{M}\mathbf{x} + \mathbf{c}$) or given by a set of polynomial relations of this form $\phi : \mathbf{x}(\in K^n) \mapsto \mathbf{y}(\in K^n)$:

$$\begin{aligned} y_1 &= x_1; \\ y_2 &= x_2 + q_2(x_1); \\ y_3 &= x_3 + q_3(x_1, x_2); \\ \vdots\ \vdots &\qquad \vdots \\ y_n &= x_n + q_n(x_1, x_2, \ldots, x_{n-1}); \end{aligned}$$

the indices of x_i can be permuted. Basic properties of tame transformations are:

- We can compute the preimage $\mathbf{x} = \phi^{-1}(\mathbf{y})$ as easily as $\mathbf{y} = \phi(\mathbf{x})$; but
- it is difficult to write $\mathbf{x}$ explicitly as a function of $\mathbf{y}$. As we solve for each x_i serially, the degree of the polynomials expressing x_i in y_j escalate quickly — exponentially, even if most q_k's are merely quadratic:

$$\begin{aligned} x_1 &= y_1; \\ x_2 &= y_2 - q_2(x_1) = y_2 - q_2(y_1); \\ x_3 &= y_3 - q_3(x_1,\, x_2) = y_3 - q_3(y_1,\, y_2 - q_2(y_1)); \\ \vdots\ \vdots &\qquad \vdots \\ x_n &= y_n - q_n(x_1, \ldots,\, x_{n-1}) \\ &= y_n - q_n(y_1, y_2 - q_2(y_1),\, \ldots,\, y_{n-1} - q_{n-1}(\cdots)). \end{aligned}$$

In this paper we loosely call a map *tame-like* or just *tame* when it is either a tame transformation, or if it retains, at least where it matters, the property of having *at least one* preimage with easy serial computation through substitution or the solution of only linear equations but not an explicit inverse of low degree. So a tame-like map can be neither injective nor surjective.

In general the verification map of any TTS scheme is $V : \mathrm{GF}(2^8)^n \to \mathrm{GF}(2^8)^m$ where $n > m$ and use only a single tame map. In contrast TTM, the family of public-key encryption scheme that is companion to TTS, uses two tame maps. We illustrate with an example first.

2.1 A Toy Example

We use a verification map $V = \phi_3 \circ \phi_2 \circ \phi_1 : \mathrm{GF}(2)^5 \to \mathrm{GF}(2)^3$ composed thusly:

$$\begin{array}{ccc} \phi_3 & \phi_2 & \phi_1 \\ \begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} = \mathsf{M}_3 \begin{bmatrix} y_2 \\ y_3 \\ y_4 \end{bmatrix} + \mathbf{c}_3 & \begin{aligned} y_2 &= x_2 + a_2 x_0 x_1 \\ y_3 &= x_3 + a_3 x_1 x_2 \\ y_4 &= x_4 + a_4 x_2 x_3 \end{aligned} & \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \mathsf{M}_1 \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} + \mathbf{c}_1 \end{array}$$

Normally, in GF(2^8), we choose arbitrarily the nonzero parameters a_i. Here each a_i has to be 1. We can pick any $\mathbf{c}_1$, and the invertible matrices M_1 and M_3, but we will compute a $\mathbf{c}_3$ so that all the constant terms vanish. Suppose

$$\mathbf{c}_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}; \quad \mathsf{M}_1 = \begin{bmatrix} 1\,0\,0\,1\,1 \\ 1\,1\,0\,1\,0 \\ 1\,0\,1\,0\,0 \\ 1\,1\,1\,1\,1 \\ 0\,1\,0\,1\,0 \end{bmatrix}; \quad \mathsf{M}_3 = \begin{bmatrix} 1\,1\,1 \\ 1\,0\,1 \\ 1\,1\,0 \end{bmatrix}.$$

We compose them to get $\mathbf{c}_3 = (0, 1, 0)$ and this $\mathbf{z} = V(\mathbf{w})$ (note: $w_i^2 = w_i$ in GF(2)).

$$z_0 = w_0 + w_1 + w_2 + w_3 + w_0w_1 + w_0w_2 + w_1w_3 + w_1w_4 + w_2w_4 + w_3w_4;$$
$$z_1 = w_2 + w_4 + w_0w_3 + w_1w_2 + w_1w_3 + w_1w_4 + w_2w_3 + w_2w_4 + w_3w_4;$$
$$z_2 = w_0 + w_2 + w_0w_2 + w_0w_3 + w_0w_4 + w_1w_2 + w_1w_3 + w_1w_4 + w_2w_3 + w_3w_4.$$

These quadratic polynomials form our verification function or public key. The private key would be the a_i's, $\mathbf{c}_1$, $\mathbf{c}_3$, and the inverses M_1^{-1} and M_3^{-1}. Suppose a mini-hash value $\mathbf{z} = (1, 1, 0)$ is given, we will compute a set of $\mathbf{w} = (w_0, w_1, w_2, w_3, w_4)$ that satisfies these equations. We can compute the signature $S(\mathbf{z}) = \phi_1^{-1}(\phi_2^{-1}(\phi_3^{-1}(\mathbf{z})))$ by solving three sequential sets of solutions:

1. $\mathbf{y} = \mathsf{M}_3^{-1}(\mathbf{z} - \mathbf{c}_3) = (1, 1, 1)$ is straightforward[2].
2. The solution for $\mathbf{x}$ is clearly not unique, but assigning values to x_0 and x_1 will force the rest and there are four possible values for $\mathbf{x}$: $(0, 0, 1, 1, 0)$, $(0, 1, 1, 0, 1)$, $(1, 0, 1, 1, 0)$, $(1, 1, 0, 1, 1)$.
3. As $\mathbf{w} = \mathsf{M}_1^{-1}(\mathbf{x} - \mathbf{c}_1)$, we can compute all four $\mathbf{w}$ values, which turns out to be $(1, 1, 0, 1, 1)$, $(1, 0, 0, 1, 1)$, $(1, 0, 0, 0, 1)$, $(1, 1, 1, 0, 1)$. Each could be the signature attached to the message. The recipient can verify that the signature and the hash value indeed satisfy $\mathbf{z} = V(\mathbf{w})$.

In this toy example a brute-force search takes no time. However, assuming that V does not easily decompose into its original components, and that solving for $\mathbf{w}$ is hard with real-life sized parameters then we have a secure signature that cannot easily be forged. We can see also that our trapdoor provides for a very fast signing procedure, taking not much more time than two matrix multiplications.

2.2 A Generic Form for TTS

Let K be a field. The verification map $V = \phi_3 \circ \phi_2 \circ \phi_1 : K^n \to K^m$ is the public key, where $\phi_1 : \mathbf{w} \mapsto \mathbf{x} = \mathsf{M}_1\mathbf{w} + \mathbf{c}_1$ and $\phi_3 : \mathbf{y} \mapsto \mathbf{z} = \mathsf{M}_3\mathbf{y} + \mathbf{c}_3$ are invertible affine maps in K^n and K^m respectively. $\phi_2 : K^n \to K^m$ is a tame map, called the central map or the kernel, and contains a lot of parameters. $S = \phi_1^{-1} \circ \phi_2^{-1} \circ \phi_3^{-1}$,

[2] All minus signs in this section could just have been pluses because of characteristic 2.

where ϕ_2^{-1} takes *any* preimage of ϕ_2 and does not have to be deterministic, is the signing map. The information content of $(\phi_1^{-1}, \phi_2^{-1}, \phi_3^{-1})$ is the private key.

As pointed out in [31], using a bijective affine-tame-affine public map is inadvisable because the initial equations $y_1 = x_1$ and $y_2 = x_2 + ax_1^2$ represent intrinsic vulnerabilities. TTM must resort to using another tame map because a public encryption map must retain all information content. In fact, T. Moh's prototype signature scheme ([31]) also used two tame maps, but TTS was designed with a more lightweight single tame-map approach, concentrating on being a good signature scheme.

In TTS, the signing map S may add extra information that we need not preserve through the verification process V. Hence, some initial dimensions can and will be collapsed by the central map ϕ_2:

$$\phi_2 = [\text{ projection collapsing initial } (n-m) \text{ coordinates }] \circ [\text{ tame-like map }].$$

Otherwise stated, we discard the first $n-m$ coordinates after a tame or tame-like transformation. Geometrically, given a message digest, we hide an $(n-m)$-dimensional *algebraic variety* in K^n consisting of all possible digital signatures. The probability of guessing a point on the variety correctly is $\approx |K|^{-m}$, e.g., $\approx 2^{-160}$ as $K = \mathrm{GF}(2^8)$ and $m = 20$ in our proposal. Suppose our central map takes this form:

$$\begin{aligned} \phi_2 &: \mathbf{x} = (x_0, x_1, \ldots, x_{n-1}) \mapsto \mathbf{y} = (y_{n-m}, \ldots, y_{n-1}), \\ y_k &= x_k + f_k(x_0, \ldots, x_{k-1}), \quad \text{for } k = n-m, \ldots, n-1, \end{aligned}$$

where f_k's are quadratic polynomials, then ϕ_2^{-1} can be computed thus when signing:

$$\begin{aligned} x_k &= \text{random variable } r_k \text{ in } K, \quad \text{for } k = 0, \ldots, n-m-1, \\ x_k &= y_k - f_k(x_0, \ldots, x_{k-1}), \quad \text{for } k = n-m, \ldots, n-1. \end{aligned}$$

For security reasons, the tame-like maps used in current TTS deviate slightly from the form above, while retaining the basic propertes of being *flexible in hash sizes* (because n and m can be adjusted easily) and being *serially and quickly solvable in each* x_k. Note that *the polynomials* f_k *can contain many arbitrary parameters* which will be randomly chosen and incorporated into the private keys.

2.3 The Old: TTS/2, a Previously Proposed Variant

Hereafter we fix $K = \mathrm{GF}(2^8)$. The signing map is $S = \phi_1^{-1} \circ \phi_2^{-1} \circ \phi_3^{-1} : K^{20} \to K^{28}$, where $\phi_1 : \mathbf{w} \mapsto \mathbf{x} = \mathsf{M}_1\mathbf{w} + \mathbf{c}_1$ and $\phi_3 : \mathbf{y} \mapsto \mathbf{z} = \mathsf{M}_3\mathbf{y} + \mathbf{c}_3$ are invertible affine in K^{28} and K^{20} respectively, and $\phi_2 : \mathbf{x} = (x_0, x_1, \ldots, x_{27}) \mapsto \mathbf{y} =$

$(y_8, y_9, \ldots, y_{27})$ is given below:

$$\begin{aligned}
y_8 &= x_8 + a_8\, x_0\, x_7 + b_8\, x_1\, x_6 + c_8\, x_2\, x_5 + d_8\, x_3\, x_4;\\
\vdots \;&\;\vdots \quad \vdots\\
y_k &= x_k + a_k\, x_{k-8}\, x_{k-1} + b_k\, x_{k-7}\, x_{k-2} + c_k\, x_{k-6}\, x_{k-3} + d_k\, x_{k-5}\, x_{k-4};\\
\vdots \;&\;\vdots \quad \vdots\\
y_{27} &= x_{27} + a_{27}\, x_{19}\, x_{26} + b_{27}\, x_{20}\, x_{25} + c_{27}\, x_{21}\, x_{24} + d_{27}\, x_{22}\, x_{23}.
\end{aligned}$$

To Generate Keys: Randomly choose $\mathbf{c}_1 \in K^{28}$, nonzero parameters a_i, b_i, c_i, d_i in K for $8 \le i \le 27$, invertible[3] $\mathsf{M}_1 \in K^{28\times 28}$ and $\mathsf{M}_3 \in K^{20\times 20}$. Find M_1^{-1} and M_3^{-1}. Compute $\mathbf{c}_3$ so that constant terms of $V = \phi_3 \circ \phi_2 \circ \phi_1$ vanish. *The* $20 \times 28 \times (28+3)/2 = 8680$ *coefficients of* V *are the public key;* ϕ_1^{-1}, ϕ_3^{-1} *and parameters* (a_i, b_i, c_i, d_i) *form the private key* (1312 bytes).

To sign a message M: Compute the message digest $\mathbf{z} = H(M)$; compute $\mathbf{y} = (y_8, y_9, \ldots, y_{27}) = \mathsf{M}_3^{-1}(\mathbf{z} - \mathbf{c}_3)$; pick $x_0, \ldots, x_7$ randomly, then solve sequentially for each of the x_i for $i = 8 \ldots 27$; the signature is $\mathbf{w} = \mathsf{M}_1^{-1}(\mathbf{x} - \mathbf{c}_1)$.

To verify $(M, \mathbf{w})$: Compare $V(\mathbf{w}) = \phi_3 \circ \phi_2 \circ \phi_1(\mathbf{w})$ against the hash $\mathbf{z} = H(M)$.

This variant was proposed in [6]. We shall propose some changes and explain why.

2.4 The New: TTS/4, an Efficient and More Secure TTS

We use the new $\phi_2 : \mathbf{x} = (x_0, x_1, \ldots, x_{27}) \mapsto \mathbf{y} = (y_8, y_9, \ldots, y_{27})$ given by:

$$\begin{aligned}
y_8 &= x_8 + a_8\, x_0\, x_7 + b_8\, x_1\, x_4 + c_8\, x_2\, x_6 + d_8\, x_3\, x_5;\\
\vdots \;&\;\vdots \quad \vdots\\
y_k &= x_k + a_k\, x_{k-8}\, x_{k-1} + b_k\, x_{k-7}\, x_{k-4} + c_k\, x_{k-6}\, x_{k-2} + d_k\, x_{k-5}\, x_{k-3};\\
\vdots \;&\;\vdots \quad \vdots\\
y_{23} &= x_{23} + a_{23}\, x_{15}\, x_{22} + b_{23}\, x_{16}\, x_{19} + c_{23}\, x_{17}\, x_{21} + d_{23}\, x_{18}\, x_{20};\\
y_{24} &= x_{24} + a_{24}\, x_{16}\, x_{23} + b_{24}\, x_{17}\, x_{20} + c_{24}\, x_{18}\, x_{22} + d_{24}\, \boldsymbol{x_4}\, \boldsymbol{x_{24}};\\
y_{25} &= x_{25} + a_{25}\, x_{17}\, x_{24} + b_{25}\, x_{18}\, x_{21} + c_{25}\, \boldsymbol{x_4}\, x_{23} + d_{25}\, \boldsymbol{x_5}\, \boldsymbol{x_{25}};\\
y_{26} &= x_{26} + a_{26}\, x_{18}\, x_{25} + b_{26}\, \boldsymbol{x_4}\, x_{22} + c_{26}\, \boldsymbol{x_5}\, x_{24} + d_{26}\, \boldsymbol{x_6}\, \boldsymbol{x_{26}};\\
y_{27} &= x_{27} + a_{27}\, \boldsymbol{x_4}\, x_{26} + b_{27}\, \boldsymbol{x_5}\, x_{23} + c_{27}\, \boldsymbol{x_6}\, x_{25} + d_{27}\, \boldsymbol{x_7}\, \boldsymbol{x_{27}}.
\end{aligned}$$

Some particular design features (items 2, 4, 5 are new TTS security enhancements):

[3] Usually by LU decomposition, even though it does not yield all nonsingular matrices.

1. If we write $y_k = x_k + \mathbf{x}^T F_k \mathbf{x}$, there is no canonical (symmetric) form for F_i since char $K = 2$, but the matrix $F_k + F_k^T$ is unique. Here $F_k + F_k^T$ *has rank 8 for every k* because no x_i appear twice in a quadratic term of the same equation.
2. The final 4 equations deviate from the general form; where x_{19}, x_{20}, x_{21}, x_{22} would be expected, the variables x_4, x_5, x_6, x_7 are substituted so that *at least one index in each quadratic term will be between* 4 *and* 18 *inclusive.* We will explain why in Secs. 4.2, and 4.3.
3. The rules are set up such that all eighty quadratic terms $x_i x_j$ are distinct.
4. The formula of y_k is different from the initial proposed form (TTS/2 of the previous section) in [6]. The indices in a quadratic term differ by 2, 3, 4, or 7 instead of 1, 3, 5, or 7. Principally, this is to avoid a separation of the x_i into even and odd indexed parts (see Sec. 4.5).
5. The last four equations has its corresponding x_i in a quadratic term. However, *the entire collection of relations is still a tame-like map.* The reason is

$$(1 + d_k x_{k-20}) x_k = y_k + (\text{function in } (x_0, \ldots, x_{k-1})) \text{ for } k = 24 \cdots 27.$$

   Since x_4, x_5, x_6, and x_7 are random numbers independent of the message digest, we pick them so that $1 + d_{24}x_4$, $1 + d_{25}x_5$, $1 + d_{26}x_6$, and $1 + d_{27}x_7$ are all non-zero[4] which ensures that $x_{24} \cdots x_{27}$ are easily solvable. See Sec. 2.5 and 4.4 for the reason for this change in design.

To generate a key pair and to verify a signed message $(M, \mathbf{w})$: Unchanged from the above section.

To sign a message M: Compute digest $\mathbf{z} = H(M)$; compute $\mathbf{y} = (y_8, y_9, \ldots, y_{27}) = \mathsf{M}_3^{-1}(\mathbf{z} - \mathbf{c}_3)$; pick $x_0, \ldots, x_7$ randomly such that $x_k \neq d_{k+20}^{-1}$ for $k = 4 \cdots 7$ (see item 5 above), then sequentially solve for x_i (for $i = 8 \ldots 27$); the signature is $\mathbf{w} = \mathsf{M}_1^{-1}(\mathbf{x} - \mathbf{c}_1)$, release $(M, \mathbf{w})$.

2.5 *Raison d'etre*: Avoiding a Common Kernel

In [7, 8] Coppersmith *et al* exploited a sequence of decreasing kernels in one of Shamir's Birational Permutation schemes, which meant that kernels of all quadratics in the public key must share a common intersection (see Sec. 4.4). The TTS/2 of [6] has a mild vulnerability of a similar type.

Proposition 1. *The kernels of all twenty quadratic polynomials of a TTS/2 public key intersect in a one-dimensional subspace which will yield x_{27} to the attacker.*

Proof. Take the symmetric matrix corresponding to
$y_8 = x_8 + a_8 x_0 x_7 + b_8 x_1 x_6 + c_8 x_2 x_5 + d_8 x_3 x_4$.

[4] A good way to visualize the theme is to think of $x_4, \ldots, x_7$ as *variable constants* and $1 + d_k x_{k-20}$ as *variable coefficient* of x_k for $k = 24 \cdots 27$, that is made part of the signature.

We see that no matter how the quadratic part of y_8 is written as $(\mathbf{x}^T Q\mathbf{x})$, the matrix $(Q+Q^T)$ will be as shown to the right, and that its kernel is $x_0 = x_1 = \cdots = x_7 = 0$. Indeed, it is easy to see that if a quadratic has the form $x_a x_b + x_c x_d + \cdots$ with all the indices $a, b, c, d, \ldots$ distinct from each other, $\{\mathbf{x} :\ 0 = x_a = x_b = x_c = x_d = \cdots\}$ will be the kernel of the corresponding symmetric matrix, hence $\{\mathbf{x} :\ x_{k-8} = \cdots = x_{k-1} = 0\}$ will be the kernel of the quadratic part of y_k written in $\mathbf{x}$.

$$\left[\begin{array}{cccccccc|c}
0 & 0 & 0 & 0 & 0 & 0 & 0 & a_8 & \\
0 & 0 & 0 & 0 & 0 & 0 & b_8 & 0 & \\
0 & 0 & 0 & 0 & 0 & c_8 & 0 & 0 & \\
0 & 0 & 0 & 0 & d_8 & 0 & 0 & 0 & \mathbf{0}^{20\times 8} \\
0 & 0 & 0 & d_8 & 0 & 0 & 0 & 0 & \\
0 & 0 & c_8 & 0 & 0 & 0 & 0 & 0 & \\
0 & b_8 & 0 & 0 & 0 & 0 & 0 & 0 & \\
a_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
\hline
\multicolumn{8}{c|}{\mathbf{0}^{8\times 20}} & \mathbf{0}^{20\times 20}
\end{array}\right]$$

We will take Q_k and $\hat{Q}_k$ to be the matrices relating the quadratic portions of each z_k to $\mathbf{w}$ and $\mathbf{x}$ respectively. Since z_k's are full-rank linear combinations of the y_k's, we know that $\cap_{j=0}^{19} \ker(\hat{Q}_k + \hat{Q}_k^T) = \mathrm{span}([0, \ldots, 0, 1]^T)$ because in the intersection subspace each of the $x_i = 0$ except for x_{27}, which does not appear in any quadratic term of the TTS/2 kernel. But we also know the quadratic portion of z_k to be simultaneously $\mathbf{x}^T \hat{Q}_k \mathbf{x}$ and $\mathbf{w} Q_k \mathbf{w}$, hence we have $Q_k = \mathsf{M}_1^T \hat{Q}_k \mathsf{M}_1$, and the kernels are related by $\ker(Q_k + Q_k^T) = \mathsf{M}_1^{-1}(\ker(\hat{Q}_k + \hat{Q}_k^T))$. Thus, we have $\cap_{j=0}^{19} \ker(Q_k + Q_k^T) = \mathrm{span}\left(\mathsf{M}_1^{-1}[0, \ldots, 0, 1]^T\right)$, so we have the last column of M_1^{-1} up to some factor.

Note: In TTS/4, the same reasoning find $\bigcap_{k=8}^{27} \ker \hat{Q}_k$'s to intersect in only the $\mathbf{0}$ vector.

One may argue that one column does not completely shatter TTS/2, and indeed it does not seem to be so easy to eliminate any other variable, but we should still avoid such a problem if possible. For other design rationales in TTS/4, please see Secs. 4.2 and 4.5.

3 Performance Evaluation

These days one usually turns to a better compiler, a better computer, or special-purpose hardware like Motorola's *AltiVec* (see [29]) for more performance. However, a better algorithm never hurts. We tabulate in Table 1 the better of timings from the vendor-submitted and NESSIE's own binaries for all 5 second round NESSIE digital signature candidates[5], normalized to a Pentium III at 500MHz from Table 37 in [35]. We also timed TTS/4 (written in reasonably portable C) on a PIII/500.

Since our programs were mainly proof-of-concept with clarity and ease of maintenance as the goal, we did not optimize to the hilt and compiled with the old `gcc-2.96`. With the newer `gcc3` or Intel's `icc`, better programming and aggressive optimizations, we estimate that TTS/4 can be at least 1.5× to 2×

[5] NESSIE eventually recommended RSA-PSS, ECDSA and SFLASH^{v2}.

Scheme	Signature	Pub. Key	Priv. Key	Keys Setup	Signing	Verifying
RSA-PSS	1024 bits	128 B	320 B	2.7 sec	84 ms	2.0 ms
ECDSA	326 bits	48 B	24 B	1.6 ms	1.9 ms	5.1 ms
ESIGN	1152 bits	145 B	96 B	0.21 sec	1.2 ms	0.74 ms
QUARTZ	128 bits	71.0 kB	3.9 kB	3.1 sec	11 sec	0.24 ms
SFLASH^{v2}	259 bits	15.4 kB	2.4 kB	1.5 sec	2.8 ms	0.39 ms
TTS/2	224 bits	8.6 kB	1.3 kB	5.3 ms	35 μs	0.13 ms
TTS/4	224 bits	8.6 kB	1.3 kB	5.3 ms	36 μs	0.13 ms

Table 1. TTS and NESSIE round 2 candidates signature schemes on a 500MHz Pentium III

faster. Still, TTS[6] performed credibly against the NESSIE candidates, and we note that:

- Multivariate schemes are fundamentally identical during verification — just substituting into polynomials. Here, TTS is faster than its multivariate relatives SFLASH^{v2} ([42]) and QUARTZ ([41]) on due to its smaller dimensions. Of course, it can also be somewhat inconvenient if the dimensions are not divisible by 4, as in the FLASH family's $K^{26} \rightarrow K^{37}$.
  Signing is determined by the kernel (middle quadratic portion) action. QUARTZ is slow since it has to solve high-order equations. A FLASH-like scheme is faster. Any TTS variant runs especially fast because inverting a tame-like map is simple and straightforward.
- Comparison with RSA-PSS, ECDSA or ESIGN is a different kettle of fish altogether. Multivariate schemes[7] is a lot faster than the more traditional contenders.
  Here, we have the modular exponentiation in RSA, a well-understood problem painstakingly optimized by over a quarter-century of computer science. Yet SFLASH^{v2} is faster and TTS/4 even more pronouncedly so. Clearly, there are intrinsic advantages to multivariate PKC. Due to concerns of security (see [34]), NESSIE recommends RSA-1536 with higher exponents as opposed to RSA-1024 and $e = 3$, which surely would further cut down the speed by at least 50% or more without special hardware. This decision can be considered at least somewhat vindicated with news of recent advances (e.g. [44]) on factoring large numbers.
- While TTS/4 (and in general TTS) does very well speedwise, unfortunately (although predictably) it is not the best in every category. All variants of TTS suffer from that common bugbear of multivariate cryptography: large keys. ECDSA is the undisputed king in small key sizes, although it uses discrete logarithms and is also slower than the multivariates.
  Thankfully, now smart cards can have on-card storage upward of 32kB. The TTS public keys, while not so trim, is smaller than that of its more robust French cousins and tolerable at 8.6kB. It was mentioned by the authors of SFLASH ([42]) that another choice is to have the private key on card

[6] TTS/2 and TTS/4 are almost identical with TTS/4 needing four extra lookups.
[7] With the exception of the slow signing action of QUARTZ.

and be able to spit out the public key when needed; the same holds true for TTS/4. *Using this technique, we are able to fit TTS key generation and signing functions on a very low-end 8051 device with only 8kB of ROM and no extra RAM. The signing time is 133 ms and total key set up time around one minute. This is rather better than SFLASH*^{v2}.

A note about future extensibility. Eventually the world will move to 256-bit hashes, and a rough estimate is that an analog to TTS/4 will use about 2.5 times as many CPU cycles; in contrast, a RSA-based scheme gets to use longer hashes for free; ECDSA will be hurt grievously when forced up to a longer hash; SFLASH will probably scale slightly worse than a TTS-based scheme. All told, we expect TTS type schemes to be speed-competitive 1-2 generations of hashes down the road. We experimented with various constructions which showed that we can make security scale as needed (Sec. 4.2).

4 Cryptanalysis

Solving generic quadratic equation systems is NP-hard ([18]), so barring an inherent flaw in TTS, there should be no sub-exponential time algorithms. But we should still take note of essentially brute-force solutions because some practical-sized systems may be solved that way, and in reasonable time. Other ideas, some ingenious, are used to cryptanalyze other multivariate schemes. We examine all ideas known to us but each seems to fail against TTS/4 without substantial modification.

First we deal with the more general approaches, designed to be applicable against *all multivariate signature schemes*, and describe in order the state of the art methods of both brute force searches (Sec. 4.1) and the more circumspect techniques of linearization (Sec. 4.2) and Gröbner bases (Sec. 4.3), and how each functions against TTS/4. Then we complements the above content by discussing attacks applicable to particular multivariate signature schemes.

4.1 Search Methods

The most advanced search methods against multivariate quadratic signature schemes ([9]) known to us so far were presented at PKC 2002. We summarize and test each given method in [9] against the current TTS/4. In each method, the aim is to solve m quadratic equations in $(w_i)_{i=1\ldots n}$ over GF(q).

Algorithm A The general idea is as follows: Pick $2k$ equations and divide the variables into groups of k and $n-k$. Separate the quadratic equations into crossterms involving variables of both groups and quadratics dependent in each group of variables only, i.e. without loss of generality:

$$z_i = g_i(w_1, \ldots, w_k) + \sum_{j=1}^{k} w_j \cdot \left(\sum_{\ell=k+1}^{n} \beta_{ij\ell} w_\ell \right) + h_i(w_{k+1}, \ldots, w_n).$$

We impose $2k^2$ linear relations $\sum_{\ell=k+1}^{n} \beta_{ij\ell} w_\ell = \gamma_{ij}$ on the variables w_{k+1}, $\dots$, w_n. If $n \geq 2k(k+1)$ and $m \geq 2k$, then we can find $\bar{k} = (n-k) - 2k^2 \geq k$ independent variables $\bar{w}_1, \dots, \bar{w}_{\bar{k}}$ such that $h_i(w_{k+1}, \dots, w_n) = h'_i(\bar{w}_1, \dots, \bar{w}_{\bar{k}})$. The equations become

$$g_i(w_1, \dots, w_k) + \sum_{j=1}^{k} \gamma_{ij} w_j = z_i - h'_i(\bar{w}_1, \dots, \bar{w}_{\bar{k}}).$$

By evaluating the left side for all possible q^k combinations and storing the results, then evaluating the right side for all $q^{\bar{k}}$ combinations, i.e. using a birthday attack and trading space for time, this can be solved in $\approx 2q^{\bar{k}} k\bar{k}^2$ time instead of $q^{k+\bar{k}} k\bar{k}^2$. The extra factor is of course the number of multiplications needed to evaluate $2k$ polynomials in $\bar{k}$ variables.
The upshot is the search will take $\approx q^{-k}$ as long as it was originally. [9] gives the complexity as $C_A \approx q^{m-k}$ where $k = \min(m/2, \left\lfloor \sqrt{n/2 - \sqrt{n/2}} \right\rfloor)$.
If we count operations per search unit, it really should be $\approx mn^2 q^{m-k}$. For TTS/4 with $(q, n, m) = (2^8, 28, 20)$, $k = 3$, and $C_A \approx 2^{151}$.

Algorithm B The general thrust is that k variables are eliminated before embarking on a brute-force search: Treat all quadratic terms $w_i w_j$ with $i, j \leq k$ as variables and eliminated first, leaving a linear system of equations in $w_1 \dots w_k$ to be solved in terms of the other variables. To give an illustrative example, in the case of TTS/4, there are 28 variables and 20 equations, and we can use 15 of the equations as a linear system to solve for the 15 quadratic terms w_0^2, $w_0 w_1, \dots$, $w_3 w_4$, w_4^2, and eliminate them from the remaining 5. Now we run a brute-force search on the variables $w_5, \dots, w_{27}$, substituting each set of values into the five equations left above and solve the resulting system for w_0, w_1, w_2, w_3, w_4 before making the consistency check. Not only have we saved the number of variables we have to guess by $k (= 5)$, instead of doing mn^2 finite field multiplications (lookups) per guess we now only have $\approx k(m-k)^2 + k^3/3$. [9] gives the complexity as $C_B \approx K \cdot q^{m-k}$, where $k = \left\lfloor \sqrt{2m+2} - 3/2 \right\rfloor$. The coefficient $K = \max(C_2, C_3)$ where C_3 is the number of operations needed to solve a system of k linear equations ($k^3/3$ multiplications) and C_2 is given as $\approx (k\,(m-k)^2)$. It appears that the authors of [9] were slightly careless in their presentation, because the requirement for k, the larger the better, is really $m - k(k+1)/2 \geq k$ so for $m = 20$ as for TTS/4, instead of $k = 4$ and $q^{m-k} = 2^{128}$ we should have $q^{m-k} = 2^{120}$. They also probably should have written $C_2 + C_3$ instead of $\max(C_2, C_3)$. Anyway, $C_B \approx 2^{130}$.

Algorithm C The general approach of this attack is to reduce into XL/FXL, but it is as inapplicable to TTS/4 just as to TTS/2 ([6]), because it requires $n \geq 2m$ ([9]).

4.2 Linearization-Type Attacks: XL and FXL

Kipnis and Shamir first introduced ([27]) *relinearization*, refining the linearization techniques usually used to solve systems of high-order polynomial equations by using relationships between monomials. The simplest variant, "degree-4 relinearization", is based on the simple fact that $(ab)(cd) = (ac)(bd) = (ad)(bc)$, in any field. Relinearization ameliorates somewhat the problem of too many extraneous solutions and is used against HFE. There are more complex higher-degree improvements, but the relinearization technique can be considered superseded by XL below (Sec. 4.2), because XL (or FXL) is expected to work in whenever relinearization does ([11]).

XL and FXL XL (and its variant FXL) can be viewed as refinements of relinearization ([11]), although both normally work with more equations than variables. The procedure at degree-D on quadratic equations (l_j) is:

1. Generate all products of arbitrary monomials of degree $D-2$ or less with each l_j; linearize by considering every monomial as an independent variable.
2. Performing Gaussian elimination on the set of equations, ordering the set of variable such that monomials in a given variable (say the first variable w_0) will always be the last to go. The remaining equation should only have terms in w_0 left.
3. Solve *a la* Berlekamp for w_0; repeat if any independent variable remains.

The FXL variant takes every possible guess at a number of variables then uses XL on the remainder. Normally, we want the number of equations to be at least one and usually 2 more than the number of variables. Another XL derivative, XL', reduce down to a system of equations in more than one variable. A recent report has it XL' can break FLASH^{v2} ([10]). Simulations supposedly show XL/FXL as effective on randomly selected quadratics — which also points to its undoing, as we shall see below.

Hilbert-Serre Theory, Solution Sets at Infinity and Why XL/FXL Fails Linearization type attacks such as XL and relinearization have a fatal weakness. They are only effective under certain circumstances, one where in a sense the set of equations is generic. In more mathematical terms, if XL can guarantee a solution, *the solution set at* ∞ *must be at most zero-dimensional.* This came from a venerable theory of Hilbert and Serre ([32]). Of course, a cryptographer will by definition be using non-generic quadratics *the infinity solution set thereof he or she can ensure to be positive-dimensional.* An extension of the argument implies that XL' is ineffective for cryptanalyzing TTS.

Let $V = \phi_3 \circ \phi_2 \circ \phi_1$ be a verification map of some TTS variant. Given a message digest $\mathbf{z}$, forging a digital signature is equivalent to finding a solution to the system of equations $\mathbf{z} = V(\mathbf{w})$. We homogenize $\mathbf{z} = V(\mathbf{w})$ in the *projective space* and let H_∞ be its *solution set at infinity.* It so happens that $\dim H_\infty$ is an important parameter for multivariate schemes because it relates directly to security under XL/FXL and Gröbner bases attacks. We claim that $\dim H_\infty \geq 12$ for TTS/4.

Since both ϕ_1 and ϕ_3 are affine and invertible, we need to consider how ϕ_2 behaves at ∞ only. We can ignore linear terms in ϕ_2, because to every non-highest-degree term is multiplied a positive power of an extra variable x_∞ during homogenization, and "at ∞" means precisely $x_\infty = 0$. Since all quadratic terms in ϕ_2 vanish when we set $x_4 = \cdots = x_{18} = 0$, there are at least the 13 free variables $x_0, \ldots, x_3, x_{19}, \ldots, x_{27}$ in this solution set, hence $\dim H_\infty \geq 13 - 1 = 12$. The claim is proved.

If the attacker successfully guess at 8 variables, the number of variables n will reduce to 20 and $\dim H_\infty$ to 4. Of course, this is not guaranteed to work! Take the example in Sec. 2.1, for all $\mathbf{w}$ in the solution set, $w_0 = w_4 = 1$. These are the *inessential* variables, and a randomly guessed assignment or restrictions on such a variable would most likely leading to a contradiction.

Part of Heisuke Hironaka's Fields Medal work was an algorithm to determine the essential variables thereof ([23]) over characteristic zero fields. Unfortunately we are working with characteristic two fields, so Hironaka's methods (using many partial derivatives) fail in this case. Absent an oracle, *the attacker now must guess which variables to fix,* adding considerably to the running time.

Assuming that 8 variables *are* guessed correctly. Now $\dim H_\infty = 4$. Over GF(2), XL/FXL can always work by including $w_i^2 = w_i$ for every i ([12]). Here, for each extra dimension in $H_\infty = 4$, the attacker needs to augment the set of equations by a number of quadratics equivalent to a Fröbenius relation $w_i^{256} = w_i$ for an essential variable w_i — maybe $w_i = p_1^2$, $p_1 = p_2^2, \ldots, p_6 = p_7^2$, $p_7 = w_i^2$. In fact the XL/FXL attacker would need to add *32 extraneous equations and 28 more variables.*

[11] claims a running time of $Aq^\mu n^{c\sqrt{n}}$ for XL/FXL, where A is the time needed to evaluate a set of polynomials, or about $mn^2/2$ multiplications; μ is the number of variables in which must be assigned by exhaustive search ("F" is for to fix); and c "the order of the Gaussian reduction algorithm", which was claimed to be $\log_2 7 \approx 2.8$. We have $n = 48$, $m = 52$ now, and at least 2^{108} complexity (times A, the amount of effort to evaluate one set of polynomials). In practice it should be a lot more.

Giving the best of all worlds to the attacker, he guess again at 4 correct variables, and succeeds in reducing $\dim H_\infty$ to zero. Even with that much luck, $n = 16$. Since $A \approx mn^2/2$, we have $A \cdot (256)^4 \cdot (16)^{2.8\times\sqrt{16}} \approx 2^{88}$. Thus TTS/4 need not worry about XL/FXL (and hence relinearization). The same kind of construction shows that if XL represents the most effective family of attacks, we can scale the TTS schemes so that they go up in complexity by 2^{16} when m goes up by 2 and n by 3.

4.3 Gröbner Bases

Gröbner Bases is a well-known way of solving polynomial equations. The classic algorithm for computing Gröbner bases, Buchberger's algorithm, involves ordering all monomials (usually lexicographically) and take some appropriate algebraic combinations of the equations to eliminate the top monomial serially, until only one variable remains and then solve for that variable (*a la* Berlekamp). This method has been extended into more powerful variants by J. Faugére, called $\mathbf{F_4}$ and $\mathbf{F_5}$ ([14, 15]). $\mathbf{F_5}/2$, an adaptation of $\mathbf{F_5}$, was used to break an HFE challenge in April 2002 ([16]).

The linearization methods of Sec. 4.2 can be considered simplified versions of Gröbner bases, and the latter are also affected by the underdeterminedness of the system. The attacker must guess at enough variables to make the program run faster. So there is the problem as is described in Sec. 4.2.

Similar to XL/FXL, Gröbner bases method is also affected by $\dim H_\infty$. But there is a difference: Since dependencies are located and collated at runtime, Gröbner bases method does not become non-functional if there is a non-zero $\dim H_\infty$. Luckily for us, it does add immensely to its time complexity.

Computing the Gröbner basis of a system of m polynomial equations of maximal degree d in n variables has a worst-case time complexity $m^3 d^{O(n^3)}$ ([4]); when the solution set is of dimension ≤ 0, this bound can be cut down to $d^{O(n^2)}$ ([5]). There is one significant theoretical exception ([28], theorem 3): *if (and essentially only if)* $\dim H_\infty \leq 0$, *we can find a Gröbner basis of degree* $\leq (d-1)n+2$ *and hence finish in* ***at most*** *time* $O(d^n)$. As a practical matter, with a suitable algorithm the exponent can be made smaller by a factor $L(q)$, where $L(2) \doteqdot 11.11$ and $L(3) \doteqdot 6.455$, but decreases quickly to 1 for larger values of q. So over *small* base fields — and this was the case for the HFE challenge 1 mentioned above — we can also finish computing a Gröbner basis a lot faster. *In general, computing a Gröbner basis takes time square-exponential in* n *when* $\dim H_\infty > 0$. *Its actual running time is very hard to estimate, but we have expert opinions to the effect that ([2]) for large fields and practical dimensions, Gröbner Bases are currently unlikely to be effective as an attack.* Therefore, for large base fields such as GF(2^8), current Gröbner-based methods cannot be used to cryptanalyze well-designed multivariate schemes, e.g., the current TTS variants, effectively.

4.4 The Coppersmith Attack vs Shamir's Birational Permutations Schemes

Shamir proposed a family of birational permutation signature schemes in [43], but soon afterwards Coppersmith *et al* found a successful attack ([7]). One specific case[8] attacked by Coppersmith *et al*, "sequentially linearized birational permutations", has $y_1 = x_1$, and $y_k = \ell_k(x_1, \ldots, x_{k-1})x_k + q_k(x_1, \ldots, x_{k-1})$ for $k = 2 \cdots n$ with ℓ's linear and q's homogeneously quadratic. Take two invertible $\mathbb{Z}_N$ square matrices ($N = pp'$ with p, p' prime), and transform $\mathbf{x}$ to $\mathbf{w}$, $(y_2, \ldots, y_n)$ to $\mathbf{z}$. The private key is the ℓ's, the $q's$, and the two invertible matrices; the user lets $\mathbf{z}$ be the message digest and finds $(y_2, \ldots, y_n)$, assigns a random x_1, solves sequentially for the rest of $\mathbf{x}$, then finds the signature $\mathbf{w}$. The public key is the quadratic forms giving $\mathbf{z}$ in $\mathbf{w}$.

[8] We invert Shamir's notations to be parallel to ours.

Sketch of attack: take the symmetric matrices M_j of y_j considered as quadratic forms of $\mathbf{x}$. These have a decreasing sequence of kernels in $\mathbf{x}$-space (ditto their images in $\mathbf{w}$-space) that we will try to find. Take λ_i such that the characteristic polynomial for $\bar{z}_i = z_i - \lambda_i z_n$ has a double root. Run recursively on the $\bar{z}_i$, which all have at least the kernel of M_{n-1}. We will have found a set of quadratic forms that are essentially equal to the original ones and enables us to forge signatures.

One can only admire the ingenuity of Coppersmith, Stern, and Vaudenay in looking for common kernel spaces. Theobald took pains to issue a similar warning ([45]) that "varying ranks of quadratic forms" may cause security concerns. Thankfully a TTS designer can arrange for a kernel without an onion-like sequence of decreasing kernels. Still, we consider TTS/4 to be partly inspired by their work.

4.5 Separation of Oil and Vinegar

In a simplified illustrative example of Patarin's *Oil and Vinegar* signature scheme, the private key is an invertible matrix $A \in K^{2n\times 2n}$ over a finite field K and n matrices $F_j \in K^{2n\times 2n}$ with zeroes in all of the upper left quarter $n \times n$ entries. The signer releases as the public key the matrices $G_j \equiv A^T F_j A$. To sign, take the message digest to be $(m_1, \ldots, m_n) \in K^n$. Assign random variables to the last n components ("vinegar") of $\mathbf{y}$, and solve the equations $\mathbf{y}^T F_j \mathbf{y} = m_j$ for the first n components ("oil") of $\mathbf{y} \in K^{2n}$. Since each F_j has its upper left quarter zero, the equations are linear in the oil variables, and $\mathbf{x} = A^{-1}\mathbf{y}$ is the signature, verifiable via $\mathbf{x}^T G_j \mathbf{x} = m_j$.

Here each F_j maps the subspace with $y_{n+1} = y_{n+2} = \cdots = y_{2n} = 0$ ("oil" subspace) to the subspace ("vinegar") $y_1 = \cdots = y_n = 0$. The cryptanalysis by Kipnis and Shamir builds on the corollary that each $F_j^{-1}F_i$ maps the "oil" subspace to itself, and each $G_j^{-1}G_i$ shares an eigenspace (the image of the oil subspace under A) for suitable (i, j). This eigenspace can be determined, enabling forged signatures. See ([26]) for details on how K-S attacked Patarin's original, more complete scheme.

"Unbalanced" Oil and Vinegar ([25]) aims to patch this by using more vinegar variables, but must tread a fine line: Too few vinegar variables, and there is still a successful reduction; too many, and brute force attacks of [9] (see Sec. 4.1) works. No such concerns exist in TTS/4. In TTS/2 we can see a separation of the variables into even and odd portions, and in TTS/4 we can also apportion the variables into $x_4, x_5, \ldots, x_{18}$ and $x_0, \ldots, x_3, x_{19}, \ldots, x_{27}$. But there are fundamental differences to [25, 26]:

1. In TTS, the vinegar (freely assigned) variables are $x_0, \ldots, x_7$. Suppose an attacker finds the common eigenspaces for the TTS/2. Where in OV or UOV he would have decomposed the signature space into dependent "Oil" and independent "Vinegar" components, here he finds himself with two identical and mutually codependent Creamy Italian portions of 4 vinegar to 10 oil (the x_i with even and odd indices respectively). The same successful determination

for TTS/4 will result in dependent Vinaigrette ($x_0,\ldots,\ x_3$ plus $x_{19},\ldots,\ x_{27}$) and independent Ranch ($x_4,\ x_6,\ldots,\ x_{18}$) portions, neither of which seems particularly useful to him.

2. According to the analysis in [25], with more dependent "oil" variables than independent "vinegar" variables the attack of [26] carries over without modification[9], but not with more vinegar than oil. The latter is the case for TTS/4 with more variables in Ranch than in Vinaigrette.

4.6 Other Special Attacks

This includes MinRank, attacks on 2R and SFLASH, and IP-based attacks:

The MinRank attack: The MinRank attack ([22]) is a type of "intelligent brute-force approach". Unfortunately the authors' presentation contained some errors ([33]), so while the idea may be valuable it would not function against TTS/4 without extensive revision. As stated in ([6]), TTS/2 was written so that every y_i has the same high rank of 8 and can be easily adjusted upwards just in case MinRank can be made to run. TTS/4 retains this property. *We have constructed variants of TTS with just a few bytes more in private keys, and 10% more of running time that has rank of 10 or 12. This is sufficient to settle any possibles problems with MinRank.*

Attacks against SFLASH: The FLASH family of public-key digital signature schemes ([42]) are based on C^{*--} ([40]), Patarin's adaptation of C^* to avoid his own ([37]) attack. The original SFLASH scheme used a subfield of GF(2) for all coefficients in its private and public keys. Gilbert and Minier observed ([21]) this to be a vulnerability and attacked the original SFLASH successfully, but their attack affects neither the current SFLASH^{v2} nor TTS/4.
Geiselmann *et al* observed ([19, 20]) that the middle portion of any FLASH variant is *homogeneous of degree two,* and showed that to be a vulnerability. Their attack gives constant parts of both affine mappings cheaply. However, it does not entirely break SFLASH and is inapplicable to a Tame Transformation type method like TTS/4 since the tame portion has linear terms.

Attacks against 2R schemes: The recent "2-round schemes" proposed by Patarin drew lots of fire ([3, 46, 47]), but the general structure of 2R schemes were so different from TTS that it is improbable for any of the suggested attacks to function against TTS/4 without substantial modification.

Patarin's IP approach: If all the parameters in the tame portion of TTS were to be fixed, then the security of TTS/4 will depend on the difficulty of the IP problem [38, 39]. But TTS/4 should not have to fear from an IP-based attack. The central portion of Tame Transformation based methods contain lots of parameters (in the private key). This approach therefore will be hard to patch to be working against any TTS variant.

[9] We find that hard to put into practice since the G_j's taken as square matrices are not invertible.

5 Conclusions

Multivariate Public-Key Cryptography is clearly a burgeoning research area rich in surprises and new discovery. For example, we saw that trivial changes to the structure in the previous TTS formulations can make an attacker's life harder and the scheme more secure, and made adjustments accordingly. We do not doubt that there shall be further attacks against multivariate schemes, attacks against the TTS genre and even specific attacks tailored against TTS/4, but we are confident that the myriad variations possible in the structure of tame and tame-like maps means that TTS will adapt and survive in the wilderness as a family of secure and fast signature schemes. In summary:

> The just-proposed TTS/4 seems efficacious and impervious to known attacks. Tame Transformations, literally the centerpiece of TTS, seem to have many good properties required of a low-degree birational permutation without its drawbacks. A principal advantage is that the central quadratic portion of the scheme — a tame-like map — is easily mutable, variable with many parameters, nonhomogeneous, and very fast.

We feel justified in stating that the TTS family merits further attention.

Acknowledgements

The authors are indebted to an anonymous referee whose incisive comments provided the necessary insight to correct an oversight that would have been very costly indeed.

The first author would like to thank Professor T. Moh for his encouragement and assistance. The second author would like to thank his faithful Ping for her support and tolerance of his eccentricities during the work leading up to this manuscript.

References

[1] M. Akkar, N. Courtois, R. Duteuil, and L. Goubin, *A Fast and Secure Implementation of SFLASH*, PKC 2003, LNCS v. 2567, pp. 267–278. 321

[2] M. Bardet, J.-C. Faugére, and B. Salvy, *Complexity of Gröbner Basis Computations for Regular Overdetermined Systems,* preprint and private communication. 333

[3] E. Biham, *Cryptanalysis of Patarin's 2-Round Public Key System with S Boxes (2R)*, Eurocrypt 2000, LNCS v. 1807, pp. 408–416. 335

[4] L. Caniglia, A. Galligo, and J. Heintz, *Some New Effectivity Bounds in Computational Geometry*, AAECC-6, 1988, LNCS v. 357, pp. 131–151. 333

[5] L. Caniglia, A. Galligo, and J. Heintz, *Equations for the Projective Closure and Effective Nullstellensatz*, Discrete Applied Mathematics, 33 (1991), pp. 11-23. 333

[6] J.-M. Chen and B.-Y. Yang, *Tame Transformation Signatures with Topsy-Turvy Hashes*, proc. IWAP '02, Taipei. 325, 326, 330, 335

[7] D. Coppersmith, J. Stern, and S. Vaudenay, *Attacks on the Birational Permutation Signature Schemes*, Crypto'93, lncs v. 773, pp. 435–443. 321, 326, 333

[8] D. Coppersmith, J. Stern, and S. Vaudenay, *The Security of the Birational Permutation Signature Schemes*, Journal of Cryptology, 10(3), 1997, pp. 207–221. 321, 326

[9] N. Courtois, L. Goubin, W. Meier, and J. Tacier, *Solving Underdefined Systems of Multivariate Quadratic Equations*, PKC 2002, lncs v. 2274, pp. 211–227. 329, 330, 334

[10] N. Courtois, L. Goubin, and J. Patarin, *SFLASH^{v3}, a Fast Asymmetric Signature Scheme*, preprint available at `http://eprint.iacr.org/2003/211`. 331

[11] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt 2000, lncs v. 1807, pp. 392–407. 331, 332

[12] N. Courtois and J. Patarin, *About the XL Algorithm over GF(2)*, CT-RSA 2003, lncs v. 2612, pp. 141–157. 332

[13] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Trans. Info. Theory, vol. IT-22, no. 6, pp. 644-654. 320

[14] J.-C. Faugére, *A New Efficient Algorithm for Computing Gröbner Bases (F4)*, Journal of Pure and Applied Algebra, 139 (1999), pp. 61–88. 333

[15] J.-C. Faugére, *A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5)*, Proceedings of ISSAC, ACM Press, 2002. 333

[16] J.-C. Faugére and A. Joux, *Algebraic Cryptanalysis of Hidden Field Equations (HFE) Cryptosystems Using Gröbner Bases*, Crypto 2003, lncs v. 2729, pp. 44-60. 333

[17] H. Fell and W. Diffie, *Analysis of a Public Key Approach Based on Polynomial Substitution*, Crypto'85, lncs v. 218, pp. 340–349. 321

[18] M. Garey and D. Johnson, *Computers and Intractability, A Guide to the Theory of NP-completeness*, 1979, p. 251. 329

[19] W. Geiselmann, R. Steinwandt, and T. Beth, *Attacking the Affine Parts of SFLASH*, 8th International IMA Conference on Cryptography and Coding, lncs v. 2260, pp. 355–359. 335

[20] W. Geiselmann, R. Steinwandt, and T. Beth, *Revealing 441 Key Bits of* sflash^{v2}, Third NESSIE Workshop, 2002. 335

[21] H. Gilbert and M. Minier, *Cryptanalysis of SFLASH*, Eurocrypt 2002, lncs v. 2332, pp. 288–298. 335

[22] L. Goubin and N. Courtois, *Cryptanalysis of the TTM Cryptosystem*, Asiacrypt 2000, lncs v. 1976, pp. 44–57. 335

[23] H. Hironaka, *Resolution of singularities of an algebraic variety over a field of characteristic zero, Parts I and II,* Annals of Mathematics, 79 (1964), pp. 109-203, 205-326. 332

[24] H. Imai and T. Matsumoto, *Algebraic Methods for Constructing Asymmetric Cryptosystems*, AAECC-3, lncs v. 229, pp. 108–119. 321

[25] A. Kipnis, J. Patarin, and L. Goubin, *Unbalanced Oil and Vinegar Signature Schemes*, Crypto'99, lncs v. 1592, pp. 206–222. 334, 335

[26] A. Kipnis and A. Shamir, *Cryptanalysis of the Oil and Vinegar Signature Scheme*, Crypto'98, lncs v. 1462, pp. 257–266. 334, 335

[27] A. Kipnis and A. Shamir, *Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization*, Crypto'99, lncs v. 1666, pp. 19–30. 331

[28] D. Lazard, *Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations*, EUROCAL '83, lncs v. 162, pp. 146–156. 333

[29] B. Lucier, *Cryptography, Finite Fields, and AltiVec*, `http://www.simdtech.org/apps/group_public/download.php/22/Cryptography.pdf` 327

[30] T. Matsumoto and H. Imai, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, Eurocrypt'88, lncs v. 330, pp. 419–453. 321

[31] T. Moh, *A Public Key System with Signature and Master Key Functions*, Communications in Algebra, 27 (1999), pp. 2207–2222. 320, 321, 324

[32] T. Moh, *On The Method of XL and Its Inefficiency Against TTM*, available at `http://eprint.iacr.org/2001/047` 331

[33] T. Moh and J.-M. Chen, *On the Goubin-Courtois Attack on TTM*, available at `http://eprint.iacr.org/2001/072` 335

[34] *NESSIE Security Report, V2.0*, available at `http://www.cryptonessie.org` 328

[35] *Performance of Optimized Implementations of the NESSIE Primitives, V2.0*, available at
`http://www.cryptonessie.org` 321, 327

[36] H. Ong, C. Schnorr, and A. Shamir, *A Fast Signature Scheme Based on Quadratic Equations*, 16th ACM Symposium on Theory of Computations, 1984, pp. 208-216. 321

[37] J. Patarin, *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*, Crypto'95, lncs v. 963, pp. 248–261. 321, 335

[38] J. Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, Eurocrypt'96, lncs v. 1070, pp. 33–48. 335

[39] J. Patarin, L. Goubin, and N. Courtois, *Improved Algorithms for Isomorphism of Polynomials*, Eurocrypt'98, lncs v. 1403, pp. 184–200. 335

[40] J. Patarin, L. Goubin, and N. Courtois, *C^*_{-+} and HM: Variations Around Two Schemes of T. Matsumoto and H. Imai*, Asiacrypt'98, lncs v. 1514, pp. 35–49. 335

[41] J. Patarin, N. Courtois, and L. Goubin, *QUARTZ, 128-Bit Long Digital Signatures*, CT-RSA 2001, lncs v. 2020, pp. 282–297. Available at `http://www.cryptonessie.org`. 328

[42] J. Patarin, N. Courtois, and L. Goubin, *FLASH, a Fast Multivariate Signature Algorithm*,
CT-RSA 2001, lncs v. 2020, pp. 298–307. Also `http://www.cryptonessie.org`. 328, 335

[43] A. Shamir, *Efficient Signature Schemes Based on Birational Permutations*, Crypto'93, lncs v. 773, pp. 1–12. 321, 333

[44] A. Shamir and E. Tromer, *Factoring Large Numbers with the TWIRL Device*, Crypto 2003, lncs v. 2729, pp. 1-26. 328

[45] T. Theobald, *How to Break Shamir's Asymmetric Basis*, Crypto'95, lncs v. 963, pp. 136–147. 334

[46] D. Ye, Z. Dai, and K. Lam, *Decomposing Attacks on Asymmetric Cryptography Based on Mapping Compositions*, Journal of Cryptology, 14(2), 2001, pp. 137–150. 335

[47] D. Ye, K. Lam, and Z. Dai, *Cryptanalysis of "2R" Schemes*, Crypto'99, lncs v. 1666, pp. 315–325. 335

Note: Full version @ IACR e-Print archive
`http://eprint.iacr.org/2003/160`.

An Efficient Revocation Algorithm in Group Signatures

Zewen Chen[1], Jilin Wang[2], Yumin Wang[2], Jiwu Huang[1], and Daren Huang[3]

[1] School of Information Science and Technology
Sun Yat-Sen Univ., Guangzhou, 510275, P. R. China
zw_chen75@hotmail.com; isshjw@zsu.edu.cn
[2] State key Lab. of Integrated Service Networks
Xidian Univ., Xi'an, Shanxi, 710071, P. R. China
jilinwang@msn.com; ymwang@xidian.edu.cn
[3] School of Mathematics and Computational Science
Sun Yat-Sen Univ., Guangzhou, 510275, P. R. China

Abstract. The problem of secure and efficient revocation of membership without incurring big costs has been considered, but no satisfactory solution was reported. In this paper, we propose a new revocation method of membership based on the ACJT group scheme. Our solution is efficient in that it only needs one multiplication and one exponentiation, which the length of exponent is fixed, to update the public key for the group manager to exclude a group member, and the signing and verifying procedure are independent of the number of current group members and excluded group members. To the best of our knowledge, the signing and verifying procedures in prior revocation schemes are dependent of the number either current group members or excluded group members, and thus the group manager needs a heavy computation load to update the public key.

1 Introduction

A group signature scheme allows a group member to anonymously sign a message on behalf of a group. The group manager has the authority to add a user into the group and to open the anonymity of a signature. The salient features of group signature make it attractive for many special applications, such as bidding [10], anonymous e-cash [8]. However, for group signature schemes to be adapted to real applications, a few problems need to be solved. Among them one of the most important things is the efficiency of revocation of membership.

The problem of revocation of membership within such a framework without incurring big costs has been considered, but until now no solution was satisfied. Bresson and Stern [3] proposed the first viable solution for revocation of membership. Unfortunately, their solution requires the signature size to be linear with respect to the number of excluded members. Moreover, it is based on the group signature scheme [6] proposed by Camenisch and Stadler, which had been found later to have certain security problems. Song [11] proposed two interesting revocation methods based on the ACJT scheme. Both methods are notable since, in addition to standard revocation, they also provide retroactive revocation as well as forward security. Moreover, they offer constant-length signature, which is an improvement over the Bresson and Stern's result. However, the verification task remains computationally intensive, and is linear in the number of excluded group members. Ateniese and Tsudik [2] proposed another solution where the size of a group signature is independent of the number of excluded group members. However, the verification task like the Song's scheme is linear in the number of excluded group members. Moreover, their solution used so-called double discrete logarithms, which results in the complexity of proving, signing and verifying to be rather high. Kim et al [7] also proposed a revocation scheme, however, it is broken. That is, excluded group members can still prove membership, because the excluded members can update their membership information in the same way as non-excluded members after the group manager changed the group's key [5]. Camenisch and Lysyanskaya [5] put forward a dynamic accumulator that accumulates primes and allows to dynamically add and delete inputs from the accumulator and applied it to revocation of membership in group signature schemes, such as the one due to Ateniese et al. [1], and efficient revocation of credentials in anonymous credential systems, such as the one due to Camenisch and Lysyanskaya [4]. Using the accumulator, the revocation did not alter the complexity of any operations of the underlying schemes. In particular, the cost of a group signature verification or credential showing increased by only a small constant factor, less than 2. Although, the scheme has so many advantages, it also exists some disadvantages: the member must perform a local computation to update his witness, which is linear in the number of changes that have taken place; the proof that a committed value was accumulate is complex, which suffers the drawback of high communication costs; the burden on the verifier is that he should look at the public key frequently, the burden on the group manager is that he should update the public key whenever there is a Join or Revoke event. G.Tsudik and S.Xu [12] also constructed a dynamic accumulator that accumulates composites and applied it in membership revocation. Though they claimed that their scheme was more efficient than [5], they suffered the same problems as [5], because they also used the accumulator. Moreover, in all above revocation schemes, the group manager at least needs one exponent computation to update the public key.

Thus, until now, all schemes have a linear dependency either on the number of current members, or on number of excluded members. As we have noted above, this linear dependency incurs some or all of the following significant cost: (1) the group manager has to do a lot of exponent computations to update the public key whenever a Join or Revoke event happens; (2) for the group member, he has to update his witness, which is linear dependency of the number of current or excluded members, to prove, as the part of signing, that his certificate is not revoked.; (3) for the verifier, he has to perform a computational test on the message received from the user for each item in the list of revoked certificates.

The main contribution of this paper is to propose a novel revocation algorithm in group signatures, which is obtained by incorporating a new building block, i.e. a protocol for proving knowledge of co-prime. Compared with all prior revocation schemes, our scheme is more efficient and practical: (1) the group manager only needs one multiplication and one exponentiation, which the length of exponent is fixed, to update the public key for excluding a group member, while in prior revocation schemes, at least one exponent computation is needed to update the public value and the length of exponent may be linear in the number of excluded group members; (2) the signing and verifying procedures are independent of the number of current and excluded members, while in prior schemes, they are not. More concretely, whenever there is a Join or Revoke event in [5] or [12], for the group manager, this results in one or more exponentiations to update the accumulator value, and for group members, they must update their witnesses, which are linear dependency of excluded members. While in our scheme, the group manager only needs one multiplication and one exponentiation with the fixed length of exponent, to update the public key to exclude a group member and the group members do not need to update any value.

Our solution's only overhead over ACJT scheme is the following: Each member needs to do 8 times exponentiations per signature and one knowledge proof to prove that he is not an excluded member. The verifier should look the public key frequently.

The rest of this paper is organized as follows. In section 2, we describe some basic tools used in ACJT scheme, which later is exposed in section 3. In section 4, we explain our method to achieve revocation of memberships. The security properties are considered in section 5. Finally, the paper concludes in section 6.

2 Preliminaries

2.1 Number-Theoretic Assumptions

Let $G = \langle g \rangle$ is a cyclic group in which order $\#G$ is unknown and its bit-length ℓ_G is publicly known. The discrete logarithm of $y \in G$ to base g is integer $x \in Z$ such that $y = g^x$ in G. There also exists a collision-resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^k$ and a security parameter $\varepsilon > 1$.

Assumption 1. *(RSA Assumption). There exists a probabilistic algorithm T such that for all probabilistic polynomial-time algorithms A, all polynomials $p(\cdot)$, and all sufficiently large ℓ_g,*

$$Pr[z = u^e \mid (G,z,e) := T(1^{\ell_g}), u := A(G, z, e)] < 1/p(\ell_g).$$

Assumption 2. *(Strong-RSA Assumption). There exists a probabilistic algorithm T such that for all probabilistic polynomial-time algorithms A, all polynomials $p()$, and all sufficiently large ℓ_g,*

$$Pr[z = u^e \mid (G,z) := T(1^{\ell_g}),\ (u,e > 1) := A(G,z)] < 1/p(\ell_g).$$

Assumption 3. *(Decisional Diffie-Hellman Assumption). There is no probabilistic polynomial-time algorithm that distinguishes with non-negligible probability between the distributions D and R, where D=(g, g^x, g^y, g^z) with $x, y, z \in {}_R Z_{\#G}$ and R=(g, g^x, g^y, g^{xy}) with $x, y \in {}_R Z_{\#G}$.*

2.2 Signatures of Knowledge

Many group signature schemes use the notion of signature of knowledge. The notion of signature of knowledge is based on the Schnorr digital signature scheme [9].

Knowledge of a discrete logarithm. Given an element $y \in G$, a signature of knowledge of the discrete logarithm of y to base g on the message m is a pair $(c, s) \in \{0,1\}^k \times \pm\{0,1\}^{\varepsilon(\ell_G+k)+1}$ satisfying

$$c = H(m\|y\|g\|g^s y^c)$$

Such a pair can be computed by a prover who knows the secret value x as follows. He chooses a random value $r \in \pm\{0,1\}^{\varepsilon(\ell_G+k)}$ and computes c and s as:

$$c = H(m\|y\|g\|g^r), s := r - xc$$

Knowledge of equality of two discrete logarithms. Let y_1, y_2, g, $h \in G$, a signature of knowledge of equality of two discrete logarithms of both y_1 to base g and y_2 to base h on the message m is a pair $(c, s) \in \{0,1\}^k \times \pm\{0,1\}^{\varepsilon(\ell_G+k)+1}$ satisfying

$$c = H(m\|y_1\|y_2\|g\|h\|g^s y_1^c\|h^s y_2^c)$$

The party possessing the secret x is able to compute the signature by choosing a random value $r \in \{0,1\}^{\varepsilon(\ell_G+k)}$ and then computing c and s as:

$$c = H(m\|y_1\|y_2\|g\|h\|g^r\|h^r), s := r - cx$$

Knowledge of discrete logarithm lying in a given interval. Given y, $g \in G$, a signature of knowledge of a discrete logarithm of y to base g lying in $[X - 2^{\varepsilon(\ell+k)}, X + 2^{\varepsilon(\ell+k)}]$ on the message m is a pair $(c, s) \in \{0,1\}^k \times \pm\{0,1\}^{\varepsilon(\ell_G+k)+1}$ satisfying

$$c = H(m\|y\|g\|g^{s-cX} y^c)$$

¿From the knowledge of $x \in [X - 2^\ell, X + 2^\ell]$, this signature is obtained by choosing a random value $r \in \pm\{0,1\}^{\varepsilon(\ell_G+k)}$ and then computing c and s as:

$$c = H(m\|y\|g\|g^r), s := r - c(x - X)$$

3 The ACJT Scheme

In this section we provide an overview of the ACJT scheme [1]. The ACJT scheme is proven secure and coalition- resistant under the strong RSA and DDH assumptions. The security of the non-interactive group signature scheme relies additionally on the random oracle model.

Let $\varepsilon > 1$, k and ℓ_p are security parameters and define two integer intervals as follows:

$$\Delta = [2^{\lambda_1} - 2^{\lambda_2}, 2^{\lambda_1} + 2^{\lambda_2}], \Gamma = [2^{\gamma_1} - 2^{\gamma_2}, 2^{\gamma_1} + 2^{\gamma_2}]$$

where $\lambda_1 > \varepsilon(\lambda_2 + k) + 2$, $\lambda_2 > 4\ell_p$,$\gamma_1 > \varepsilon(\gamma_2 + k) + 2$ and $\gamma_2 > \lambda_1 + 2$.

In the initial phase, the group manager (GM) sets the group public key and his secret key.

Setup:

- GM chooses random secret ℓ_p bit primes p', q' such that $p = 2p'+1, q = 2q'+1$ are prime. Set the modulus $n = pq$.
- GM chooses random elements $a, a_0, g, h \in {}_R QR(n)$.
- GM chooses a random secret element $x \in Z_{p'q'}$, and sets $y = g^x \bmod n$.
- The group public key is: Y=(n, a, a_0, y, g, h).
- The corresponding secret key is $S = (p', q', x)$.

Suppose that a new user wants to join the group. We assume that communication between the user and the group manager is secure, i.e., private and authentic.

Join:

- UserP_i generates a secret $\tilde{x}_i \in [0, 2^{\lambda_2}]$, a random integer $\tilde{r} \in [0, n^2]$ and sends $C_1 = g^{\tilde{x}_i} h^{\tilde{r}}$ to GM and proves that C_1 is formed correctly.
- GM checks that $C_1 \in QR(n)$. If this the case, GM selects α_i and $\beta_i \in [0, 2^{\lambda_2}]$ at random and sends (α_i, β_i) to P_i.
- P_i computes $x_i = 2^{\lambda_1} + (\alpha_i \tilde{x}_i + \beta_i \bmod 2^{\lambda_2})$ and sends the value $C_2 = a^{x_i} \bmod n$ to GM. P_i also proves to GM:
  - **a)** that the discrete logarithm of C_2 to base a lies in Δ and
  - **b)** knowledge of integers u, v, w such that (1) u lies in $[-2^{\lambda_2}, 2^{\lambda_2}]$, (2) u equals the discrete logarithm of $C_2/a^{2^{\lambda_1}}$to base a, (3) $C_1^{\alpha_i} g^{\beta_i}$ equals $g^u (g^{2^{\lambda_2}})^v h^w$.
- GM checks that $C_2 \in QR(n)$, if this is the case and all the above proofs are correct. GM selects a prime $e_i \in \Gamma$, computes$A_i := (C_2 a_0)^{1/e_i}$and sends P_i the membership certificate $[A_i, e_i]$.
- User P_i verifies that $a^{x_i} a_0 = A_i^{e_i} \bmod n$.

Thereafter, the new member can generate a group signature as follows:

Sign:

– Generate a random value $w \in_R \{0,1\}^{2\ell_p}$ and compute:

$$T_1 = A_i y^w \bmod n, \quad T_2 = g^w \bmod n, \quad T_3 = g^{e_i} h^w \bmod n$$

– Randomly choose

$$r_1 \in \pm\{0,1\}^{\varepsilon(\gamma_2+k)},\ r_2 \in \pm\{0,1\}^{\varepsilon(\lambda_2+k)},$$

$$r_3 \in \pm\{0,1\}^{\varepsilon(\gamma_1+2\ell_p+k+1)}, \text{ and } r_4 \in \pm\{0,1\}^{\varepsilon(2\ell_p+k)}$$

and then compute:
a) $d_1 = T_1^{r_1}/(a^{r_2}y^{r_3}), d_2 = T_2^{r_1}/g^{r_3}, d_3 = g^{r_4}, d_4 = g^{r_1}h^{r_4}$
b) $c = H(g\|h\|y\|a_0\|a\|T_1\|T_2\|T_3\|d_1\|d_2\|d_3\|d_4\|m)$
c) $s_1 = r_1 - c(e_i - 2^{\gamma_1}), s_2 = r_2 - c(x_i - 2^{\lambda_1}), s_3 = r_3 - ce_iw, s_4 = r_4 - cw$
– Output $(c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$.

A group signature is basically a signature of knowledge of (1) a value $x_i \in \Delta$ such that $a^{x_i}a_0$ is ElGamal-encrypted in (T_1, T_2) under y and of (2) the e_i-th root of that encrypted value.

A verifier checks the validity of a signature $(c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$ on message m as follows:

Verify:

– Compute:

$$c' = H(g\|h\|y\|a_0\|a\|T_1\|T_2\|T_3\|a_0^c T_1^{s_1-c2^{\gamma_1}}/(a^{s_2-c2^{\lambda_1}}y^{s_3}) \bmod n\| \\ T_2^{s_1-c2^{\gamma_1}}/g^{s_3} \bmod n\|T_2^c g^{s_4} \bmod n\|T_3^c g^{s_1-c2^{\gamma_1}}h^{s_4} \bmod n\|m)$$

– Accept the signature if and only if

$$c = c', s_1 \in \pm\{0,1\}^{\varepsilon(\lambda_2+k)+1}, s_2 \in \pm\{0,1\}^{\varepsilon(\lambda_2+k)+1},$$

$$s_3 \in \pm\{0,1\}^{\varepsilon(\lambda_1+2\ell_p+k+1)+1}, s_4 \in \pm\{0,1\}^{\varepsilon(2\ell_p+k)+1}$$

In the event that the signer must be subsequently identified (e.g., in case of a dispute) GM executes the following procedure:

Open:

– Check the signature's validity via the verify procedure.
– Recover A_i as $A_i = T_1/T_2^x$.
– Prove that $\log_g^y = \log_{T_2}^{(T_1/A_i \bmod n)}$.

4 Achieving Revocation

In this section, we propose a solution to exclude membership from a group without leaking any useful information about their past signature. In case of member exclusion, the group manager would issue a product of e_j's corresponding to all the excluded members. Any user could continue to sign if he is able to prove that his e_i contained in the signature is relatively prime to that product. The group manager needs only one multiplication and one exponentiation, which the length of exponent is fixed, to update the public key for excluding a group member. The signing and verifying procedures are independent of the number of the current and excluded group members.

4.1 Proving Co-prime

Suppose that there exists a group $\tilde{G} =< \tilde{g} >=< \tilde{h} >$ of order $\varphi(n)$ and a publicly known number E. If the prover, who does not know $\varphi(n)$ and the discrete logarithm of $\tilde{h}$ to base $\tilde{g}$, wants to prove that a number e is relatively prime to E without revealing e. The protocol for proving knowledge of co-prime is as follows.

Prover:

– Computes the integers a,b such that ae+bE=1 by the extended GCD algorithms.
– Randomly chooses w_1, w_2,w_3
– Computes

$$y_1 = \tilde{g}^{w_1}, y_2 = \tilde{g}^{w_2}, y_3 = \tilde{g}^{w_3}, y_4 = \tilde{h}^{-(aw_1+w_2+w_3)},$$
$$T_1 = \tilde{g}^e\tilde{h}^{w_1}, T_2 = T_1^a\tilde{h}^{w_2}, T_3 = (\tilde{g}^E)^b\tilde{h}^{w_3}.$$

– Randomly chooses $r_1, r_2, r_3, r_4, r_5, r_6, r_7$
– Computes

$$R_1 = \tilde{g}^{r_1}, R_2 = \tilde{g}^{r_2}, R_3 = \tilde{g}^{r_3}, R_4 = \tilde{h}^{r_4}, R_5 = \tilde{g}^{r_5}\tilde{h}^{r_1},$$
$$R_6 = T_1^{r_6}\tilde{h}^{r_2}, R_7 = (\tilde{g}^E)^{r_7}\tilde{h}^{r_3}$$

– Computes

$$c = H(\tilde{g}\|\tilde{h}\|E\|y_1\|y_2\|y_3\|y_4\|T_1\|T_2\|T_3\|R_1\|R_2\|$$
$$R_3\|R_4\|R_5\|R_6\|R_7)$$

– Computes

$$s_1 = r_1 - cw_1, s_2 = r_2 - cw_2, s_3 = r_3 - cw_3, s_4 = r_4 + c(aw_1 + w_2 + w_3),$$
$$s_5 = r_5 - ce, s_6 = r_6 - ca, \quad s_7 = r_7 - cb$$

– Publishes (y_1, y_2, y_3, y_4, T_1, T_2, T_3, c, s_1, s_2, s_3, s_4, s_5, s_6, s_7).

Verifier (anyone):

– Computes

$$c' = H(\tilde{g}\|\tilde{h}\|E\|y_1\|y_2\|y_3\|y_4\|T_1\|T_2\|T_3\|\tilde{g}^{s_1}y_1^c\|\tilde{g}^{s_2}y_2^c\|\tilde{g}^{s_3}y_3^c\|$$
$$\tilde{h}^{s_4}y_4^c\|\tilde{g}^{s_5}\tilde{h}^{s_1}T_1^c\|T_1^{s_6}\tilde{h}^{s_2}T_2^c\|(\tilde{g}^E)^{s_7}\tilde{h}^{s_3}T_3^c)$$

– Verifies $c = c$. If the verification fails, stop the verification process.
– If $\tilde{g} = T_2T_3y_4$, then e is relatively prime to E.

Using the notation introduced by Camenisch and Stadler [6]. The above proof can be denoted by

$$PK1\{(\alpha,\beta,\gamma,\vartheta,\tau,\xi,\zeta) : y_1 = \tilde{g}^{\vartheta}\Lambda y_2 = \tilde{g}^{\tau}\Lambda y_3 = \tilde{g}^{\xi}\Lambda y_4 = \tilde{h}^{\zeta}\Lambda$$
$$T_1 = \tilde{g}^{\alpha}\tilde{h}^{\vartheta}\Lambda T_2 = T_1^{\beta}\tilde{h}^{\tau}\Lambda T_3 = (\tilde{g}^E)^{\gamma}\tilde{h}^{\xi}\Lambda\tilde{g} = T_2T_3y_4\}$$

Lemma 1. *Under discrete logarithm and RSA assumptions, the* $PK1\{(\alpha,\beta,\gamma,\vartheta,\tau,\xi,\zeta) : y_1 = \tilde{g}^{\vartheta}\Lambda y_2 = \tilde{g}^{\tau}\Lambda y_3 = \tilde{g}^{\xi}\Lambda y_4 = \tilde{h}^{\zeta}\Lambda T_1 = \tilde{g}^{\alpha}\tilde{h}^{\vartheta}\Lambda T_2 = T_1^{\beta}\tilde{h}^{\tau}\Lambda T_3 = (\tilde{g}^E)^{\gamma}\tilde{h}^{\xi}\Lambda\tilde{g} = T_2T_3y_4\}$ *can only be produced by the prover who uses a number which is relatively prime to E.*

Proof. Let d=gcd(α, E) and A be an attacker who executes the $PK1$ protocol with $d \neq 1$. Then we will show that A can break RSA assumption, i.e., find a s, such that $ds = 1 \bmod \varphi(n)$.

¿From $y_4 = \tilde{h}^{\zeta}, T_1 = \tilde{g}^{\alpha}\tilde{h}^{\vartheta}, T_2 = T_1^{\beta}\tilde{h}^{\tau}, T_3 = (\tilde{g}^E)^{\gamma}\tilde{h}^{\xi}$ and $\tilde{g} = T_2T_3y_4$, A gets $\tilde{g} = T_2T_3y_4 = \tilde{g}^{\alpha\beta+E\gamma}\tilde{h}^{\beta\vartheta+\tau+\xi+\zeta}$. Because the prover does not know the discrete logarithm of $\tilde{h}$ to the base $\tilde{g}$. There must be $\tilde{g} = \tilde{g}^{\alpha\beta+E\gamma}$ under the discrete logarithm assumption. That is, $\alpha\beta + E\gamma = 1 \bmod \varphi(n)$. Let $s = \alpha\beta/d + E\gamma/d$. Then $ds = 1 \bmod \varphi(n)$. This concludes the proof. □

Theorem 1. *Under the RSA and strong RSA assumptions, the interactive protocol corresponding to above PK1 protocol is a knowledge proof of the fact that the prover has a number* α *which is relatively prime to E.*

In theorem 1, the $PK1$ protocol is to show that given values y_1, y_2, y_3, y_4, T_1, T_2, T_3 and the expression $\tilde{g} = T_2T_3y_4$, the prover uses (α, β, γ, θ, τ, ξ, ζ) such that $y_1 = \tilde{g}^{\vartheta}, y_2 = \tilde{g}^{\tau}, y_3 = \tilde{g}^{\xi}, y_4 = \tilde{h}^{\zeta}, T_1 = \tilde{g}^{\alpha}\tilde{h}^{\vartheta}, T_2 = T_1^{\beta}\tilde{h}^{\tau}, T_3 = (\tilde{g}^E)^{\gamma}\tilde{h}^{\xi}$ and α is relatively prime to E.

Proof. To show that the protocol is a knowledge proof, we have to show that the knowledge extractor is able to recover the α which is relatively prime to E once it has found two accepting tuples. Let

$$(y_1, y_2, y_3, y_4, T_1, T_2, T_3, c, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$$

and

$$(y_1, y_2, y_3, y_4, T_1, T_2, T_3, c', s_1', s_2', s_3', s_4', s_5', s_6', s_7')$$

be two accepting tuples.

Because $R_1 = \tilde{g}^{s_1} y_1^c = \tilde{g}^{s_1'} y_1^{c'}$, we have $y_1^{c-c'} = \tilde{g}^{s_1'-s_1}$. Let $d_1 = \gcd(c-c', s_1'-s_1)$, using the extended Euclidean algorithm, we can obtain the values u_1, v_1 such that $u_1(c-c')+v_1(s_1'-s_1) = d_1$ and hence we have $\tilde{g} = \tilde{g}^{(u_1(c-c')+v_1(s_1'-s_1))/d_1} = (\tilde{g}^{u_1} y_1^{v_1})^{(c-c')/d_1}$. If $d_1|c-c$ then $\tilde{g}^{u_1} y_1^{v_1}$ is a $(c-c)/d_1$ root of $\tilde{g}$. Since this contradicts Strong RSA assumption, we have $d_1 = c-c$. Hence we can compute the integer $\vartheta = (s_1' - s_1)/(c-c')$ such that $y_1 = \tilde{g}^{\theta}$.

For $R_5 = \tilde{g}^{s_5} \tilde{h}^{s_1} T_1^c = \tilde{g}^{s_5'} \tilde{h}^{s_1'} T_1^{c'}$, this can be rewritten as $\tilde{g}^{s_5'-s_5} \tilde{h}^{s_1'-s_1} = T_1^{c-c'}$. Because $\vartheta = (s_1'-s_1)/(c-c')$, we have $\tilde{g}^{s_5'-s_5} = (T_1 \tilde{h}^{-\vartheta})^{c-c'}$. Similar as the proof of above, we obtain $\alpha = (s_5'-s_5)/(c-c')$ such that $T_1 = \tilde{g}^{\alpha} \tilde{h}^{\vartheta}$.

Likewise, from $R_2 = \tilde{g}^{s_2} y_2^c = \tilde{g}^{s_2'} y_2^{c'}$ and $R_6 = T_1^{s_6} \tilde{h}^{s_2} T_2^c = T_1^{s'_6} \tilde{h}^{s'_2} T_2^{c'}$ we obtain β, τ such that $T_2 = T_1^{\beta} \tilde{h}^{\tau} = \tilde{g}^{\alpha\beta} \tilde{h}^{\beta\vartheta+\tau}$. Also from $R_3 = \tilde{g}^{s_3} y_3^c = \tilde{g}^{s_3'} y_3^{c'}$, $R_7 = (\tilde{g}^E)^{s_7} h^{s_3} T_3^c = (\tilde{g}^E)^{s_7'} h^{s_3'} T_3^{c'}$, we are able to get γ, ξ such that $T_3 = (\tilde{g}^E)^{\gamma} \tilde{h}^{\xi}$.

Therefore, $\tilde{g} = \tilde{g}^{\alpha\beta+E\gamma} \tilde{h}^{\zeta}$ from $\tilde{g} = T_2 T_3 y_4$. From lemma 1, we can obtain $\alpha\beta + E\gamma = 1$, so α is relative prime to E. This concludes the proof. □

We say the verifier can not get any useful information about α because the verifier only get $(y_1, y_2, y_3, y_4, T_1, T_2, T_3, c, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$, if he wants to get α, β, γ, then he has to solve the discrete logarithms problem.

In the above protocol, the value E is in Z and it may be very large, so if we apply above method directly in membership revocation, the computation of $\tilde{g}^E$ may become inefficient. In order to achieve the efficient membership revocation, we will make little modification of the above $PK1$ protocol for proving knowledge of co- prime, i.e., using the $g_E = \tilde{g}^{E \bmod \varphi(n)}$ instead of $\tilde{g}^E$ in the above protocol. Here, g_E is a public value, computed by someone knowing the $\varphi(n)$. Similar as above, we get the new protocol for proving knowledge of co-prime, denoted by

$$PK2\{(\alpha,\beta,\gamma,\vartheta,\tau,\xi,\zeta) : y_1 = \tilde{g}^{\vartheta} \varLambda y_2 = \tilde{g}^{\tau} \varLambda y_3 = \tilde{g}^{\xi} \varLambda y_4 = \tilde{h}^{\zeta} \varLambda$$
$$T_1 = \tilde{g}^{\alpha} \tilde{h}^{\vartheta} \varLambda T_2 = T_1^{\beta} \tilde{h}^{\tau} \varLambda T_3 = (g_E)^{\gamma} \tilde{h}^{\xi} \varLambda \tilde{g} = T_2 T_3 y_4\}$$

Lemma 2. *Under discrete logarithm and RSA assumptions, the* $PK2\{(\alpha,\beta,\gamma,\vartheta,\tau,\xi,\zeta) : y_1 = \tilde{g}^{\vartheta} \varLambda y_2 = \tilde{g}^{\tau} \varLambda y_3 = \tilde{g}^{\xi} \varLambda y_4 = \tilde{h}^{\zeta} \varLambda T_1 = \tilde{g}^{\alpha} \tilde{h}^{\vartheta} \varLambda T_2 = T_1^{\beta} \tilde{h}^{\tau} \varLambda T_3 = (g_E)^{\gamma} \tilde{h}^{\xi} \varLambda \tilde{g} = T_2 T_3 y_4\}$ *can only be produced by the prover who uses a number which is relatively prime to* E.

Proof. Let d=gcd(α, E) and A be an attacker who executes the $PK2$ protocol with $d \neq 1$. Then we will show that A can break RSA assumption, i.e., find a s, such that $ds = 1 \bmod \varphi(n)$.

¿From $y_4 = \tilde{h}^{\zeta}, T_1 = \tilde{g}^{\alpha} \tilde{h}^{\vartheta}, T_2 = T_1^{\beta} \tilde{h}^{\tau}, T_3 = (g_E)^{\gamma} \tilde{h}^{\xi}$ and $\tilde{g} = T_2 T_3 y_4$, A gets $\tilde{g} = T_2 T_3 y_4 = \tilde{g}^{\alpha\beta+\gamma(E \bmod \varphi(n))} \tilde{h}^{\beta\vartheta+\tau+\xi+\zeta}$. Because the prover does not know the discrete logarithm of $\tilde{h}$ to the base $\tilde{g}$. There must be $\tilde{g} = \tilde{g}^{\alpha\beta+\gamma(E \bmod \varphi(n))}$ under the discrete logarithm assumption. So, $\alpha\beta+\gamma(E \bmod \varphi(n)) = 1 \bmod \varphi(n)$. That is, $\alpha\beta + \gamma E = 1 \bmod \varphi(n)$, Let $s = \alpha\beta/d + E\gamma/d$. Then $ds = 1 \bmod \varphi(n)$. This concludes the proof. □

Theorem 2. *Under the RSA and strong RSA assumptions, the interactive protocol corresponding to above PK2 protocol is a knowledge proof of the fact that the prover has a number* α *which is relatively prime to* E.

The proof of this theorem is the same as theorem 1, so we omit it here.

4.2 Application in Revocation of Membership

Next, we will show how to apply the previous technique to construct a revocation mechanism in ACJT scheme. Please note that all elements are in G of order $\phi(n)$ and not in $QR(n)$. This does not influence the security of ACJT scheme.

The group signature scheme consists of the following procedures: setup, join, revoke, sign, verify and open. In this section, the setup, join and open procedure are the same as ACJT scheme. So we just describe the revoke, sign and verify procedure as follows.

Revoke: suppose that E_{delete}:=$\{\mathrm{G}_1,\mathrm{G}_2,\ldots,\mathrm{G}_m\}$ are current excluded group member. The group manager computes E:=$e_{G1}\ldots\ e_{Gm}$, publishes E and $g_E = g^{E \bmod \varphi(n)}$, where e_{Gi} is the prime corresponding to the G_i's certificate. When there are several group members to be excluded, let E' the product of e_{Gj}'s to be excluded. The group manager only need update the public key $E := EE'$ and use the new E to compute $g_E = g^{E \bmod \varphi(n)}$.

Sign: For the group member, who is not an excluded group member, if he wants to sign a message, he must prove that he holds the certificate $(A_i,\ e_i,x_i)$. This can be achieved by applying ACJT scheme. The group member also needs to prove that e_i is not in E, i.e., e_i is relatively prime to E. This can be obtained by theorem 2. The membership proof can be obtained as follows:

– Computes the integers a,b such that ae_i+bE=1 by the extended GCD algorithm.
– Generates random value $u, v, w \in_R \{0,1\}^{2\ell_p}$ and computes:

$$T_1 = A_i y^u, T_2 = g^u, T_3 = g^{e_i} h^u, T_4 = g^v, T_5 = T_3^a h^v, T_6 = g^w,$$
$$T_7 = (g_E)^b h^w, T_8 = h^{-(au+v+w)}$$

– Generates

$$PK3\{(\alpha,\beta,\delta,\varepsilon) : a_0 = T_1^{\alpha}(1/a)^{\beta}(1/y)^{\delta} \varLambda T_2 = g^{\varepsilon} \varLambda 1 = T_2^{\alpha}(1/g)^{\delta} \varLambda$$
$$T_3 = g^{\alpha} h^{\varepsilon} \varLambda \alpha \in \varGamma \varLambda \beta \in \varDelta\}$$
$$PK4\{(\alpha,\eta,\gamma,\varepsilon,\tau,\xi,\zeta) : T_2 = g^{\varepsilon} \varLambda T_4 = g^{\tau} \varLambda T_6 = g^{\xi} \varLambda T_8 = h^{\zeta} \varLambda T_3 = g^{\alpha} h^{\varepsilon} \varLambda$$
$$T_5 = T_3^{\eta} h^{\tau} \varLambda T_7 = (g_E)^{\gamma} h^{\xi} \varLambda g = T_5 T_7 T_8\}$$

The signature of the group membership is the tuple (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, $PK3$, $PK4$).

The first protocol $PK3$ is the same as the ACJT scheme, which is to prove holding membership certificates $(A_i,\ e_i,x_i)$. The second protocol $PK4$ is a knowledge proof protocol of the fact that e_i, which is used in $PK3$, is relatively prime to E.

The verify procedure is just to check the correctness of the $PK3$, $PK4$.

From above, we can see that our scheme is more efficient than prior schemes, because the group manager needs only one multiplication and one exponentiation with the fixed length of exponent, to update the public key for excluding a group member and other group members do not need to update their certificates. The signing and verifying procedure are independent of the number of current and excluded group members.

5 Security Analysis

In this section, we show that our proposed scheme satisfies all security properties required by a group signature scheme and that, in addition, the group manager can safely exclude group members.

Theorem 3 (1). *Under the Strong-RSA assumption, a group certificate* (A_i, e_i, x_i) *with* $x_i \in \Delta, e_i \in \Gamma$ *can be generated only by the group manager provided that the number* K *of certificates the group manager issues is polynomially bounded.*

Theorem 4 (1). *Under the strong RSA assumption, the interactive protocol corresponding to PK3 is a statistical zero-knowledge proof of knowledge of* (A_i, e_i, x_i).

Let us now discuss the security properties of the proposed group signature scheme.

Correctness: This can be achieved by verifying the correctness of $PK3$, $PK4$.

Unforgeability: If the user is not a group member and not an excluded group member, then from Theorem 3, he can't produce (A_i, e_i, x_i) such that $a_0 a^{x_i} = A_i^{e_i}$. If the user is an excluded group member, he has (A_i, e_i, x_i), where $e_i | E$. From Theorem 2, he can't prove that e_i is relatively prime to E. That is, only group members are able to sign messages on behalf of the group.

Anonymity: Given a valid group signature (T_1,T_2,T_3,T_4,T_5,T_6,T_7,T_8, $PK3$, $PK4$), from Theorem 2 and Theorem 4, we know that there is no information revealed from $PK3$, $PK4$. It is infeasible to get (A_i, e_i, x_i) from (T_1,T_2,T_3,T_4,T_5,T_6, T_7,T_8) because computing discrete logarithms is assumed to be infeasible.

Exculpability: Neither a group member nor the group manager can sign on behalf of other group members: First, from Theorem 3, we know that a group member can not sign on behalf of other group members. Next, the group manager does not get any information about a group member's secret x_i apart from a^{x_i}.

Traceability: The group manager can identify the actual signer by computing $A_i = T_1/T_2^x$.

Coalition- resistance: This follows from Theorem 2 and Theorem 3.

Unlinkability: Because no information is revealed from $PK3$, $PK4$ and (T_1, T_2, T_3, T_4, T_5,T_6, T_7, T_8) are unconditionally binding with random number u, v, w. so deciding different valid signatures were computed by the same group member is computationally hard. Following, we show that even when the group member is excluded, i.e., his e_i is published, other members cannot link his past signature. Suppose that (T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8) and $(T'_1, T'_2, T'_3, T'_4, T'_5, T'_6, T'_7, T'_8)$ are two signatures. When e_i is published, other members may know a and b such that $ae_i + bE = 1$. If they want to decide whether the two signatures are signed by the same member, they must decide whether the discrete logarithms $\log_y^{T_1/T'_1}$, $\log_g^{T_2/T'_2}$ and $\log_g^{T_3/T'_3}$ are equal or $\log_h^{T_5 T_3'^{a}/(T'_5 T_3^a)}$ and $\log_g^{T_4/T'_4}$ are equal or $\log_g^{T_6/T'_6}$ and $\log_h^{T_7/T'_7}$ are equal. This is impossible under the Decisional Diffie-Hellman assumption.

Revocability: For the excluded group member with certificate (A_i,e_i), there are two possible ways to prove his membership: (1) he chooses a number e which is different from e_i and relatively prime to E and then uses e to prove his membership. This is impossible from Theorem 3; (2) he also uses the same e_i to prove his membership. This is also impossible from Theorem 2 and Lemma 2.

6 Conclusions

In this paper, we present a new revocation method for ACJT scheme. The idea is that the group manager publishes the product of e_i's corresponding to excluded members and the legitimate group member proves that his e_j is relatively prime to the product. The new revocation scheme is more efficient than the existing schemes because the size of signature is constant and the group manager needs only one multiplication and one exponentiation with the fixed length of exponent for excluding a member. Also, the signing and verifying procedures are independent of the number of current members and excluded members. Compared with the original ACJT scheme without member exclusion, our solution's overhead is as follows: Each group member must read the public key prior to signing, then do 8 times exponentiations per signature and give a knowledge proof of co-prime. For the verifier, he should look at the public key frequently. The proposed method can be extended to obtain a credential revocation mechanism for Camenisch and Lysyanskayas's credential system [CL01].

Acknowledgement

Authors appreciate the support received from NSFC (60325208, 60172067, 60133020), "863" Program (2002AA144060), NSF of Guangdong (013164), Funding of China National Education Ministry.

References

[1] G.Ateniese, J.Camenisch, M.Joye, and G.Tsudik, A practical and provably secure coalition-resistant group signature scheme, In Advances in Cryptology- CRYPTO 2000, vol. 1880 of LNCS. pp. 255-270. Springer-Verlag, 2000

[2] G.Ateniese and G.Tsudik, Quasi-efficient revocation of group signature, http://eprint.iacr.org/2001/101/, 2001.

[3] E.Bresson and J.Stern. Group signatures with efficient revocation, In Proceedings of PKC2001, vol. 1992 of LNCS, pp. 190-206, Springer-Verlag, 2001

[4] J.Camenisch and A.Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation, Advances in Cryptology- EUROCRYPT 2001, Vol. 2045 of LNCS, pp.93-118. Springer-Verlag, 2001

[5] J.Camenisch and A.Lysyanskaya, Dynamic accumulators and application to efficient revocation of anonymous credentials, In Advances in Cryptology - CRYPTO 2002, vol. 2442 of LNCS, pp. 61-77, Springer-Verlag, 2002.

[6] J.Camenish and M.Stadler, Efficient group signatures for large groups, Proceedings of CRYPTO'97. vol. 1296 of LNCS, pp.410-424.Springer-Verlag,1997

[7] H.Kim, J.Lim and D.Lee. Efficient and secure member deletion in group signature schemes. In D.Won, editor, ICISC 2000, vol 2015 of LNCS, pp. 150-161, Springer-Verlag, 2001

[8] A.Lysyanskaya and Z.Ramzan, Group blind digital signatures: A scalable solution to electronic cash, In Financial Cryptography (FC'98), vol. 1465 of LNCS, pp. 184-197, Springer-Verlag, 1998

[9] C. P.Schnorr, Efficient identification and signature for smart cards, Crypto'89, vol 435 of LNCS, pp.239-252, Springer-Verlag, 1990

[10] K.Sakurai and S.Miyazaki, An anonymous electronic bidding protocol based on a new convertible group signature scheme, Proc. 5-th Australasian Conference on Information Security and Privacy (ACISP2000), Vol. 1841 LNCS, pp.385-399. Springer- Verlag, 2000

[11] D.Song, Practical forward secure group signature schemes, In Proceedings of 2001 ACM Symposium on Computer and Communication Security, November 2001.

[12] G.Tsudik and S.Xu. Accumulating Composites and Improved Group Signing. Asiacrypt'03, to appear

Efficient Forward and Provably Secure ID-Based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity

Sherman S.M. Chow, S.M. Yiu, Lucas C.K. Hui, and K.P. Chow

Department of Computer Science and Information Systems
The University of Hong Kong, Hong Kong
{smchow,smyiu,hui,chow}@csis.hku.hk

Abstract. Boyen [7] gave the first identity-based (ID-based) signcryption scheme that is forward secure, publicly verifiable as well as provably secure. However, his scheme aims at providing ciphertext unlinkability and anonymity which is not a desirable property in applications such as authentication of encrypted messages by firewalls [11], where any third party should be able to verify the origin of the ciphertext without knowing the content of the message and getting any help from the intended recipient. This requirement is referred as *public ciphertext authenticity*. In this paper, we give another ID-based signcryption scheme that can provide public ciphertext authenticity and is forward and provably secure as well as publicly verifiable. Our scheme is modified from Libert and Quisquater's ID-based signcryption scheme [16] and the efficiency of our scheme is comparable to other previous ID-based signcryption schemes.

Keywords. Identity-based signcryption, forward security, public verifiability, provable security, semantical security, public ciphertext authenticity, network security, firewalls, public key cryptosystems, elliptic curve cryptosystems, cryptographic primitives

1 Introduction

Identity-based (ID-based) cryptography (for examples, [20], [6] and [9]) is rapidly emerging in recent years. The distinguishing property of ID-based cryptography is that a user's public key can be any binary string, such as an email address, that can identify the user. This removes the need for senders to look up the recipient's public key before sending out an encrypted message. ID-based cryptography is supposed to provide a more convenient alternative to conventional public key infrastructure.

Confidentiality, integrity, non-repudiation and authentication are the basic requirements for many cryptographic applications. A traditional approach to achieve these requirements is to "sign-then-encrypt" the message. Signcryption scheme, introduced by Zheng in [22], is a cryptographic primitive that combines encryption and signing in one step at a lower computational cost. A recent direction is to merge the concepts of ID-based cryptography and signcryption to design efficient ID-based signcryption schemes. An ID-based signcryption scheme should provide the following properties.

Forward Security (FwSec): Following the definition from [16] and [5][1], an ID-based signcryption scheme provides forward secure *encryption* if knowing the private key of the *sender* cannot *reveal* the messages he or she signcrypted before.

Public Verifiability (PubVer): An ID-based signcryption scheme is publicly verifiable if given a message m, a signcrypted message σ, and possibly some additional information provided by the recipient, a third party can verify that σ is a valid signature of the sender for m, without knowing the recipient's private key.

Note that we adopt a different definition from that in [14], we refer the public verifiability defined in [14] as *public ciphertext authenticity*, which we will revisit later.

Provable Security (ProvSec): An ID-based signcryption scheme is said to be provably secure if it satisfies the property of *indistinguishability against adaptive chosen-ciphertext-and-identity attacks* (also known as semantical security) and is secure against *an existential forgery for adaptive chosen-message-and-identity attacks* (see Section 2.2 for formal definitions).

Depending on the applications, an ID-based signcryption scheme may need to satisfy additional requirements. In this paper, we consider the following additional requirement.

Public Ciphertext Authenticity (PubCAuth): An ID-based signcryption scheme is said to provide public ciphertext authenticity if any third party can verify the validity and the origin of the ciphertext without knowing the content of the message and getting any help from the intended recipient.

This requirement is useful in applications such as authentication of encrypted messages by firewalls [11] in which the origin of the received signcrypted message must be verified by a third party before the message is accepted in the system.

1.1 Previous Work and Our Contribution

None of the previous ID-based signcryption schemes can satisfy all the above requirements. Malone-Lee gave the first ID-based signcryption scheme [17]. His scheme provides forward security and public verifiability. However, the scheme is not semantically secure. As pointed out by [16], this scheme is the result of a combination of a simplified version of Boneh and Franklin's ID-based encryption [6] with a variant of Hess's ID-based signature [12]. Roughly speaking, the signcrypted message is a concatenation of a signature and a ciphertext. In other

[1] In [5], the concept of forward security is defined in private-key cryptography: confidentiality of data that have been encrypted using some secret information in the past is not compromised by loss of the secret at present. We borrow their definition and adopt it in public-key cryptography, where the secret information we consider here is the private signcryption key of the sender.

words, the signature of the message is visible in the signcrypted message, so the scheme cannot be semantically secure [21].

On the other hand, Nalla and Reddy's ID-based signcryption scheme [18] cannot provide public verifiability as well as public ciphertext authenticity since the verifications can only be done with the knowledge of recipient's private key. Libert and Quisquater proposed three ID-based signcryption schemes [16]. None of them can satisfy the requirements for public verifiability and forward security at the same time.

Boyen's multipurpose ID-based signcryption scheme [7] is the only existing scheme that provides public verifiability and forward security and is also provably secure. However, this scheme aims at providing ciphertext unlinkability and anonymity. So, a third party cannot verify the origin of the ciphertext, thus the scheme does not satisfy the requirement of public ciphertext authenticity. We remark that Boyen's scheme is very useful in applications that require unlinkability and anonymity.

In this paper we present another ID-based signcryption scheme that can satisfy all the above requirements. Our scheme is, in fact, modified from Libert and Quisquater's scheme. We also show that our proposed scheme is provably secure based on the assumption of the computational hardness of variants of the Decisional and Computational Bilinear Diffie-Hellman problems.

1.2 Comparison of Schemes

Table 1 shows a summary of comparing our scheme with other existing schemes in terms of the identified requirements and efficiency. We use BF+Hess to denote the simple "Encrypt-then-Sign" approach based on Boneh-Franklin's ID-based encryption scheme [6] and Hess's signature scheme [12]. We use ML, NR, XB to denote the scheme in [17], [18], [7] respectively. In [16], there are three signcryption schemes, a basic version, a modified version with shorter ciphertext, and a modified version with forward security. These three schemes are denoted as LQ-Basic, LQ-Short and LQ-Fwd respectively. Note that in BF+Hess, the ciphertext is encrypted by using the public key of the recipient, but not the private key of the sender, so the way we define forward security is not applicable and it is considered as 'N' in the table.

For efficiency, we compare the number of operations needed for the signcryption, unsigncryption, and the verification processes. We only consider the operations executed by the sender and the recipient, but not the third party as not all schemes are publicly verifiable. In fact, for third party's verification, our scheme only requires two pairing operations and one exponentiation operation. The following shows the types of operations that we consider.

1. *Pairing (Pa)*: The total number of pairing computations required. In the table, we represent this total in the form of $x(+y)$ where y is the number of operations that are independent of the message and can be pre-computed and cached for subsequent uses.

Table 1. Comparison of features and efficiency of existing signcryption schemes

	Security Requirement				Efficiency		
Schemes	FwSec	PubVer	PubCAuth	ProvSec	Pa	Mu	Ex
BF+Hess	*N*	Y	Y	*N*	2 (+ 3)	3	1
ML	Y	Y	Y	*N*	3 (+ 2)	3	1
NR[2]	Y	*N*	*N*	*N*	2 (+ 2)	2	2
LQ-Basic[3]	*N*	Y	Y	Y	5 (+ 1)	2	4
LQ-Short[3]	*N*	Y	*N*	Y	5 (+ 1)	2	4
LQ-Fwd[4]	Y	*N*	*N*	Y	3 (+ 0)	4	1
XB	Y	Y	*N*	Y	5 (+ 0)	2	2
Our Scheme	Y	Y	Y	Y	6 (+ 0)	3	0
Our Scheme's Variant	Y	Y	Y	Y	5 (+ 1)	2	2

2. *Point Multiplication (Mu)*: The total number of point multiplications required.
3. *Exponentiation (Ex)*: The total number of exponentiations required.

To summarize, our proposed scheme can satisfy all the requirements that we have identified. With pre-computed pairing result, our scheme can be executed at a cost comparable or even lower than those provably secure schemes that provide public verifiability. The rest of this paper is organized as follows. Section 2 contains some preliminaries about the framework of an ID-based signcryption scheme, the formal definitions for an ID-based signcryption scheme to be provably secure, and bilinear mapping. Our scheme is presented in Section 3. We analyze our scheme and show our scheme is provable secure in the random oracle model [4] in Section 4. Section 5 concludes the paper.

2 Preliminaries

2.1 Framework of ID-Based Signcryption Schemes

An identity-based (ID-based) signcryption scheme consists of five algorithms: `Setup`, `Extract`, `Signcrypt`, `Unsigncrypt` and `TP_Verify` (if public verifiability is satisfied). In essence, `Setup` generates common public parameters and master secret depending on the security level parameter; `Extract` generates the private key(s) for each user according to the user's public identity; `Signcrypt` produces the ciphertext from a sender to a designated recipient; `Unsigncrypt` recovers

[2] Its semantic security is not yet proved/disproved.
[3] We believe that it can be modified so that the number of Mu + Ex is reduced to 4 with 6 (+ 0) pairing operations.
[4] We believe that it can be modified so that the number of Mu + Ex can be reduced by 1.

the original message after checking its integrity and origin; `TP_Verify` enables any third party to verify the integrity and origin of the message. The functions of these algorithms are described as follows.

- `Setup`: On an unary string input 1^k where k is a security parameter, it produces the common public parameters $params$, which include a description of a finite message space together with a description of a finite ciphertext space; and the master secret s, which is kept secret by the Private Key Generator (PKG)
- `Extract`: On an arbitrary string input ID, it computes the private signcryption key S_{ID} and the private decryption key D_{ID}, corresponding to $(params, s)$. Note that in our framework, the signcryption key and the decryption key are not necessary the same.
- `Signcrypt`: On input (m, S_{ID_A}, ID_B), it outputs a signcrypted ciphertext σ, corresponding to $(params, s)$.
- `Unsigncrypt`: On input (σ, ID_A, D_{ID_B}), it outputs the original message m and ephemeral data $temp$ for public verification (if the scheme provides public verifiability), or the symbol $\perp$ if σ is not accepted as a valid ciphertext, corresponding to $(params, s)$.
- `TP_Verify`: On input $(\sigma, ID_A, m, temp)$, it outputs $\top$ for "true" or $\perp$ for "false", depending on whether σ is a valid ciphertext of message m signcrypted by ID_A or not, corresponding to $(params, s)$.

These algorithms must satisfy the standard consistency constraint of ID-based signcryption, i.e. if $\sigma =$ `Signcrypt`$((m, S_{ID_A}, ID_B)$, then we must have $(m, temp) =$ `Unsigncrypt`(σ, ID_A, D_{ID_B}) and $\top =$ `TP_Verify`$(\sigma, ID_A, m, temp)$.

2.2 Formal Security Notions

Confidentiality: Malone-Lee [17] extended the notion of semantic security for public key encryption schemes to ID-based signcryption schemes. We modify this definition slightly and our security notion is referred as *indistinguishability of identity-based signcryptions under adaptive chosen-ciphertext-and-identity attacks* (IND-IDSC-CCIA2). The similar notion has been used in [16]. Consider the following IND-IDSC-CCIA2 game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup: The challenger $\mathcal{C}$ takes a security parameter k and runs `Setup` to generate common public parameters $param$ and the master secret key s. $\mathcal{C}$ sends $param$ to $\mathcal{A}$.

Phase 1: The adversary $\mathcal{A}$ can perform a polynomially bounded number of queries in an adaptive manner (that is, each query may depend on the responses to the previous queries). The types of queries allowed are described below.

- `Extract`: $\mathcal{A}$ chooses an identity ID. $\mathcal{C}$ computes $\texttt{Extract}(ID) = (S_{ID}, D_{ID})$ and sends the result to $\mathcal{A}$.
- `Signcrypt`: $\mathcal{A}$ chooses two identities ID_i and ID_j, and a plaintext m. $\mathcal{C}$ signcrypts the plaintext by computing $\sigma = \texttt{Signcrypt}\ (m, S_{ID_i}, ID_j)$ and sends σ to $\mathcal{A}$.
- `Unsigncrypt`: $\mathcal{A}$ chooses two identities ID_i and ID_j, and a ciphertext σ. $\mathcal{C}$ computes the private decryption key D_{ID_j} by calling $\texttt{Extract}(ID_j)$, then unsigncrypts the ciphertext σ by calling $\texttt{Unsigncrypt}(\sigma, ID_i, D_{ID_j})$ and sends the resulting plaintext m or the symbol $\perp$ to $\mathcal{A}$

Challenge: The adversary $\mathcal{A}$ decides when Phase 1 ends. Then, it outputs two equal length plaintexts, m_0 and m_1, and two identities, ID_A and ID_B, on which it wishes to be challenged. The identity ID_B should not appear in any `Extract` queries in Phase 1. The challenger $\mathcal{C}$ picks a random bit b from $\{0, 1\}$, computes $\sigma = \texttt{Signcrypt}\ (m_b, S_{ID_A}, ID_B)$, and returns σ to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ can ask a polynomially bounded number of queries adaptively again as in Phase 1 with the restriction that it cannot make an `Extract` query on ID_B and cannot make an `Unsigncrypt` query on (σ, ID_A, D_{ID_B}) to obtain the plaintext for σ.

Guess: The adversary $\mathcal{A}$ has to output a guess b'. It wins the game if $b' = b$.

The *advantage* of $\mathcal{A}$ is defined as $Adv(\mathcal{A}) = |2P[b' = b] - 1|$ where $P[b' = b]$ denotes the probability that $b' = b$.

Definition 1. *An ID-based signcryption scheme is said to have the indistinguishability against adaptive chosen-ciphertext-and-identity attacks property (IND-IDSC-CCIA2 secure) if no adversary has a non-negligible advantage in the IND-IDSC-CCIA2 game.*

Notice that the adversary is allowed to make an `Extract` query on the signcrypting identity ID_A in the above definition. This condition corresponds to the stringent requirements of *insider-security* for confidentiality of signcryption [1]. On the other hand, it ensures the *forward security* of the scheme, i.e. confidentiality is preserved in case the sender's private signcryption key becomes compromised.

Unforgeability:
Again, Malone-Lee [17] extended the notion of existential unforgeability for signature schemes to ID-based signcryption schemes. The security notion in our work is referred as *existential unforgeability of identity-based signcryptions under adaptive chosen-message-and-identity attacks* (EUF-IDSC-CMIA2). Its formal definition is based on the following EUF-IDSC-CMIA2 game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup: The challenger $\mathcal{C}$ takes a security parameter k and runs `Setup` to generate common public parameters *param* and the master secret key s. $\mathcal{C}$ sends *param* to $\mathcal{A}$.

Attack: The adversary $\mathcal{A}$ can perform a polynomially bounded number of queries in an adaptive manner (that is, each query may depend on the responses to the previous queries). The types of queries allowed are described below.

– `Extract`: $\mathcal{A}$ chooses an identity ID. $\mathcal{C}$ computes `Extract`$(ID) = (S_{ID}, D_{ID})$ and sends the result to $\mathcal{A}$.
– `Signcrypt`: $\mathcal{A}$ chooses two identities ID_i and ID_j, and a plaintext m. $\mathcal{C}$ signcrypts the plaintext by computing $\sigma =$ `Signcrypt` (m, S_{ID_i}, ID_j) and sends σ to $\mathcal{A}$.
– `Unsigncrypt`: $\mathcal{A}$ chooses two identities ID_i and ID_j, and a ciphertext σ. $\mathcal{C}$ computes the private decryption key D_{ID_j} by calling `Extract`(ID_j), then unsigncrypts the ciphertext σ by calling `Unsigncrypt`(σ, ID_i, D_{ID_j}) and sends the resulting plaintext m or the symbol $\perp$ to $\mathcal{A}$

Forgery: The adversary $\mathcal{A}$ outputs (σ, ID_A, ID_B) where ID_A did not appear in any `Extract` query in the Attack phase. It wins the game if the response of the `Unsigncrypt` on (σ, ID_A, D_{ID_B}) is not equal to $\perp$.

The advantage of $\mathcal{A}$ is defined as the probability that it wins.

Definition 2. *An ID-based signcryption scheme is said to have the existential unforgeability against adaptive chosen-message-and-identity attacks property (EUF-IDSC-CMIA2 secure) if no adversary has a non-negligible advantage in the EUF-IDSC-CMIA2 game.*

Note that in the above definition, the adversary is allowed to make an `Extract` query on the forged message's recipient ID_B. Again, this condition corresponds to the stringent requirements of *insider-security* for signcryption [1], which is to ensure the non-repudiation property by preventing a dishonest user who holds a valid user's private key of the system from generating a valid ciphertext to himself/herself on other's behalf and claim the forged authenticity.

2.3 Bilinear Mapping and Bilinear Diffie-Hellman Problems

Let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \cdot)$ be two cyclic groups of prime order q. The bilinear pairing is given as $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, which satisfy the following properties:

1. *Bilinearity*: For all $P, Q, R \in \mathbb{G}_1$, $\hat{e}(P+Q, R) = \hat{e}(P, R)\hat{e}(Q, R)$, and $\hat{e}(P, Q+R) = \hat{e}(P, Q)\hat{e}(P, R)$.
2. *Non-degeneracy*: There exists $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1$.
3. *Computability*: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ $\forall P, Q \in \mathbb{G}_1$.

We assume the existence of a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the property that the following problems and their variants (see below) are hard to compute.

Definition 3. *Given two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same prime order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ and a generator P of $\mathbb{G}_1$, the Decisional Bilinear Diffie-Hellman problem (DBDHP) in $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ is to decide whether $h = \hat{e}(P,P)^{abc}$ given (P, aP, bP, cP) and an element $h \in \mathbb{G}_2$.*

Definition 4. *Given two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same prime order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ and a generator P of $\mathbb{G}_1$, the Computational Bilinear Diffie-Hellman problem (CBDHP) in $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ is to compute $h = \hat{e}(P,P)^{abc}$ given (P, aP, bP, cP).*

Both DBDHP and CBDHP are assumed to be hard and no known algorithm can solve any of them efficiently. In this paper, we consider variants of DBDHP and CBDHP, in which $c^{-1}P$ is also given as input. We refer these variants as MDBDHP (Modified DBDHP) and MCBDHP (Modified CBDHP). Obviously DBDHP and CBDHP are harder than MDBDHP and MCBDHP, respectively. However, no known existing efficient algorithm can solve MDBDHP and MCBDHP, to the best of our knowledge. Indeed, our actual scheme only needs to publish (P, aP, bP, cP). In [2], it is shown that inverse computational Diffie-Hellman problem (On input P and cP, outputs $c^{-1}P$) is equivalent to computational Diffie-Hellman problem. So it is computationally infeasible to derive $c^{-1}P$ from the public system parameters. Besides, if we consider only adaptive chosen-ciphertext attacks and existential unforgeability under adaptive chosen-message attacks, in which we do not allow the adversary to adaptively make `Extract` request; then our scheme's security is relied on DBDHP and CBDHP instead of MDBDHP and MCBDHP. We believe that MDBDHP and MCBDHP are interesting in their own rights as their intractabilities may give raise to new cryptosystems like this one, in which the private signcryption key and the private decryption key are separated.

3 The Proposed Scheme

Define $\mathbb{G}_1$, $\mathbb{G}_2$ and $\hat{e}(\cdot,\cdot)$ as in previous section. Let H_1, H_2 and H_3 be three cryptographic hash functions where $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ and $H_3 : \{0,1\}^n \times \mathbb{G}_2 \to \mathbb{F}_q^*$. Let E, D be the encryption and decryption algorithms of a secure symmetric cipher which takes a plaintext/ciphertext of length n respectively, and also a key of length n. (For example, a one-time pad cipher as used in [7].) The following shows the details of the scheme.

- `Setup`: Let P be an arbitrary generator of $\mathbb{G}_1$, the Private Key Generator (PKG) chooses $s \in \mathbb{F}_q^*$ randomly and $P_{pub} = sP$. The master-key is s, which is kept secret and known only by PKG. The system parameters are
$$\{\mathbb{G}_1, \mathbb{G}_2, q, n, P, P_{pub}, \hat{e}(\cdot,\cdot), H_1(\cdot), H_2(\cdot), H_3(\cdot,\cdot), E_{(\cdot)}(\cdot), D_{(\cdot)}(\cdot)\}$$
- `Extract`: The user with identity $ID \in \{0,1\}^*$ submits ID to PKG. PKG sets the user's public key Q_{ID} to be $H_1(ID) \in \mathbb{G}_1$, computes the user's private signcryption key S_{ID} by $S_{ID} = s^{-1}Q_{ID}$ and private decryption key by $D_{ID} = sQ_{ID}$. Then PKG sends the private keys to the user.

- `Signcrypt`: To send a message $m \in \{0,1\}^n$ to B, A follows the steps below.
  1. Choose x from $\mathbb{F}_q^*$ randomly.
  2. Compute $x_A = xQ_{ID_A}$.
  3. Compute $k_1 = \hat{e}(x_A, P)$ and $k_2 = H_2[\hat{e}(x_A, Q_{ID_B})]$.
  4. Compute $c = E_{k_2}(m)$.
  5. Compute $r = H_3(c, k_1)$.
  6. Compute $S = (x - r)S_{ID_A}$.
  7. The ciphertext is $\sigma = (c, r, S)$.
- `Unsigncrypt`: To unsigncrypt a signcrypted message (c, r, S) from A, B follows the steps below.
  1. Compute $r'_A = rQ_{ID_A}$.
  2. Compute $k'_1 = \hat{e}(S, P_{pub})\hat{e}(r'_A, P)$.
  3. Compute $k'_2 = H_2[\hat{e}(S, D_{ID_B})\hat{e}(r'_A, Q_{ID_B})]$.
  4. Recover $m = D_{k'_2}(c)$.
  5. Compute $r' = H_3(c, k'_1)$.
  6. Accept the message if and only if $r' = r$, return $\perp$ otherwise.
  7. Give (k'_2, m, σ) to third party.
- `TP_Verify`:
  1. Compute $k'_1 = \hat{e}(S, P_{pub})\hat{e}(Q_{ID_A}, P)^r$.
  2. Compute $r' = H_3(c, k'_1)$.
  3. Accept the origin of ciphertext if and only if $r = r'$.
  4. Moreover, accept the message authenticity if and only if $m = D_{k'_2}(c)$.
  5. Return $\top$ if all tests are passed, $\perp$ otherwise.

4 Analysis of the Scheme

4.1 Security

Consistency: The consistency can be easily verified by the following equations.

$$\begin{aligned}
k'_1 &= \hat{e}(S, P_{pub})\hat{e}(rQ_{ID_A}, P) \\
&= \hat{e}(xS_{ID_A}, P_{pub})\hat{e}(rS_{ID_A}, P_{pub})^{-1}\hat{e}(Q_{ID_A}, P)^r \\
&= \hat{e}(Q_{ID_A}, P)^x \\
k'_2 &= H_2[\hat{e}(S, D_{ID_B})\hat{e}(rQ_{ID_A}, Q_{ID_B})] \\
&= H_2[\hat{e}(xS_{ID_A}, D_{ID_B})\hat{e}(rS_{ID_A}, D_{ID_B})^{-1}\hat{e}(Q_{ID_A}, Q_{ID_B})^r] \\
&= H_2[\hat{e}(Q_{ID_A}, Q_{ID_B})^x]
\end{aligned}$$

Confidentiality and Forward Security: Decryption requires the knowledge of $\hat{e}(Q_{ID_A}, Q_{ID_B})^x$. For a passive adversary, the information available is only σ and k_1. It is difficult to get x from k_1 since it is difficult to invert the bilinear mapping. Only S in the signature reveals x, but it is difficult to compute x from S even with the knowledge of r and S_{ID_A} since it is difficult to compute discrete logarithm. Theorem 1 shows our scheme's confidentiality and forward security under adaptive chosen-ciphertext-and-identity attacks.

Unforgeability: Only the sender A with the knowledge of S_{ID_A} can compute S. Even with a previous valid signcrypted message of m from A, an adversary cannot make another signcrypted message m' where $m' \neq m$, since S in the signcrypted message is related to the ciphertext by $r = H_3(c, k_1)$ and the hash function is assumed to be one-way and collision-free. The security of the scheme regarding the existential unforgeability under adaptive chosen-message-and-identity attacks is given in Theorem 2.

Public Ciphertext Authenticity: Step 1 to 3 of `TP_Verify` only takes the ciphertext and the public system parameters as input and do not require the knowledge of k_2', hence the origin of ciphertext can be verified without knowing the content of messages and the help of intended recipient. As in [14], if our scheme includes the receiver's public key in the signature (by setting $r = H_3(c, k_1, Q_{ID_B})$), the *receiver* of the ciphertext is public verifiable as well.

Public Verifiability: Similar to [16], forwarding ephemeral key k_2' to any third parties convinces them that the ciphertext is the signcrypted version of a given plaintext message made by the sender (see Step 4 of `TP_Verify`), so our scheme satisfies our definition of public verifiability. Note that k_2' is just a random element computed from the public information as $k_2 = H_2[\hat{e}(Q_{ID_A}, Q_{ID_B})^x]$, where x is randomly chosen.

Provably Security: Following the ideas in [16] and [7], the following two theorems show that the proposed scheme is both IND-IDSC-CCIA2 and EUF-IDSC-CMIA2 secure.

Theorem 1 *In the random oracle model (the hash functions are modeled as random oracles), we assume that we have an adversary $\mathcal{A}$ that is able to win the IND-IDSC-CCIA2 game (i.e. $\mathcal{A}$ is able to distinguish ciphertexts given by the challenger), with an advantage ϵ when running in a time t and asking at most q_H identity hashing queries, at most q_E key extraction queries, at most q_R H_3 queries, q_R `Signcrypt` queries and q_U `Unsigncrypt` queries. Then, there exists a distinguisher $\mathcal{C}$ that can solve the MDBDH problem in $O(t + (8{q_R}^2 + 4q_U)T_{\hat{e}})$ time with an advantage*

$$Adv(\mathcal{C})^{MDBDHP(\mathbb{G}_1,P)} > \frac{\epsilon(2^k - q_U) - q_U}{q_H 2^{(k+1)}}$$

where $T_{\hat{e}}$ denotes the computation time of the bilinear map and $Adv(\mathcal{C}) = |P_{a,b,c \in \mathbb{F}_q}[1 \leftarrow \mathcal{C}(aP, bP, cP, c^{-1}P, \hat{e}(P,P)^{abc})] - P_{a,b,c \in \mathbb{F}_q, h \in \mathbb{G}_2}[1 \leftarrow \mathcal{C}(aP, bP, cP, c^{-1}P, h)]|$.

Proof. See the appendix. □

Theorem 2 *In the random oracle model (the hash functions are modeled as random oracles), we assume that we have an adversary $\mathcal{A}$ that is able to win the EUF-IDSC-CMIA2 game with an advantage $\epsilon \geq 10(q_S + 1)(q_S + q_R)q_H/(2^k - 1)$ within a time span t for a security parameter k; and asking at most q_H identity hashing queries, at most q_E key extraction queries, at most q_K H_2 queries, q_R H_3*

queries, q_S `Signcrypt` *queries and q_U* `Unsigncrypt` *queries. Then, there exists an algorithm $\mathcal{C}'$ that can solve the MCBDH problem in expected time $\leq 120686 q_R q_H 2^k t/\epsilon(2^k - 1)$.*

Proof. See the appendix. □

4.2 Against Existential Forgery of Dishonest Recipient

Since third party has no way to ensure the correctness of session key k'_2 obtained from the recipient, dishonest recipient can randomly choose k'_2 such that the signcrypted message (c, r, S) decrypts to a plaintext m' which is not equal to $D_{H_2[\hat{e}(S, D_{ID_B})\hat{e}(Q_{ID_A}, Q_{ID_B})^r]}(c)$. This issue is not addressed in previous work like [16]. A simple fix to this attack is to disclose the recipient's decryption key to the third party, but this made the scheme rather inflexible and unrealistic.

Note that this existential forgery is not really dangerous in many cases as the resulting plaintext from the decryption using a random session key k'_2 is usually unintelligible or not in a correct message format. Still we present modifications to our scheme which make our scheme secure against this attack. In the modifications, apart from the signcrypted message (c, r, S), recipient randomly chooses z from $\mathbb{F}_q^*$ and sends zD_{ID_B}, $z^{-1}S$ and zP_{pub} to the third party. This does not compromise the recipient's decryption key and only enables the third party to decrypt signcrypted messages in the form of $(c^\#, r^\#, S^\#)$ where $S^\# = S$, which is a rare case as S can be considered as randomly generated by the sender. Third party can compute a correct k'_2 by itself after checking for the correctness of these additional data (zD_{ID_B}, $z^{-1}S$ and zP_{pub}) as follows.

`TP_VerifyVariant`

1. Compute $r'_A = rQ_{ID_A}$.
2. Compute $k'_1 = \hat{e}(S, P_{pub})\hat{e}(r'_A, P)$.
3. Compute $r' = H_3(c, k'_1)$.
4. Accept the origin of ciphertext if and only if $r = r'$.
5. Check whether $\hat{e}(zD_{ID_B}, P) = \hat{e}(zP_{pub}, Q_{ID_B})$.
6. Check whether $\hat{e}(z^{-1}S, zP_{pub}) = \hat{e}(S, P_{pub})$.
7. Compute $k'_2 = H_2[e(z^{-1}S, zD_{ID_B})e(r'_A, Q_{ID_B})]$.
8. Accept the message authenticity if and only if $m = D_{k'_2}(c)$.
9. Return $\top$ if all tests are passed, $\perp$ otherwise.

4.3 Efficiency

Considering the efficiency of the proposed scheme, signcryption requires two pairing operations and two point multiplications, while unsigncryption and verification need four pairing operations and one point multiplication. For third party's verification, one exponentiation and two pairing operations are required.

As shown in table 1, our proposed scheme is the most efficient one in terms of the total number of point multiplications and exponentiations. Considering the number of pairing operations, our scheme is comparable to all of the existing schemes, in particular, for those provable secure scheme with public verifiability (without public verifiability, the scheme is an authenticated encryption scheme only rather than a signcryption scheme).

Although some researches have been done in analyzing the complexity and speeding up the pairing computation (for examples, [3], [10], [13] and [15]), pairing operations are still rather expensive. The following shows how to modify the `Signcrypt` algorithm of our scheme to make it slightly more efficient.

`SigncryptVariant`: To send a message $m \in \{0,1\}^*$ to B, A follows the steps below.

1. Choose x from $\mathbb{F}_q^*$ randomly.
2. Compute $k_1 = \hat{e}(Q_{ID_A}, P)^x$
3. Compute $k_2 = H_2[\hat{e}(Q_{ID_A}, Q_{ID_B})^x]$.
4. Compute $c = E_{k_2}(m)$.
5. Compute $r = H_3(c, k_1)$.
6. Compute $S = (x - r)S_{ID_A}$.
7. The ciphertext is $\sigma = (c, r, S)$.

In this variant, $\hat{e}(Q_{ID_A}, P)$ is pre-computed since it is independent of the message and its intended recipient, so we only need a total of five pairing operations for signcryption, unsigncryption and verification processes. Note that this modification increments the total number of point multiplications and exponentiations by one; however, the scheme is as efficient as [7] and more efficient than other existing provable secure signcryption schemes with public verifiability.

5 Conclusion

We proposed an identity-based signcryption scheme that satisfies the security requirements of forward security, public verifiability and public ciphertext authenticity. Our scheme can be shown to be provably secure under the random oracle model, with the assumption that variants of the BDH problems are hard to compute. It is an interesting problem to investigate the complexity of these variants of the BDH problems as well as to design more efficient ID-based signcryption schemes that satisfy the same set of security requirements.

References

[1] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the Security of Joint Signature and Encryption. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 83-107. Springer-Verlag Heidelberg, 2002. 357, 358

[2] Feng Bao, Robert H. Deng, and Huafei Zhu. Variations of Diffie-Hellman Problem. In Dieter Qing, Dieter Gollmann, and Jianying Zhou, editors, *Information and Communications Security, 5th International Conference, ICICS 2003, Inner-Mongolia 10-13 October, 2003, Proceedings*, volume 2836 of Lecture Notes in Computer Science. Springer, 2003. 359

[3] Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In Moti Yung, editor, *Advances in Cryptology: Proceedings of CRYPTO 2002 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18-22, 2002 Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 354-368. Springer-Verlag Heidelberg, 2002. 363

[4] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *The First ACM Conference on Computer and Communications Security*, pages 62-73, 1993. 355

[5] Mihir Bellare and Bennet Yee. Forward-Security in Private-Key Cryptography . In Marc Joye, editor, *Topics in Cryptology - CT-RSA2003, The Cryptographer's Track at RSA Conference 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 1-18, 2003. 353

[6] Dan Boneh and Matt Franklin. Identity-Based Encryption from the Weil Pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 213-229. Springer-Verlag Heidelberg, 2001. 352, 353, 354

[7] Xavier Boyen. Multipurpose Identity-Based Signcryption : A Swiss Army Knife for Identity-Based Cryptography. In Dan Boneh, editor, *23rd International Conference on Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 382-398. Springer Verlag, 2003. 352, 354, 359, 361, 363, 369

[8] Jae Choon Cha and Jung Hee Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups . In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, Sixth International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings,* volume 2567 of *Lecture Notes in Computer Science*. Springer, 2002. 369

[9] Sherman S. M. Chow, Lucas C. K. Hui, S. M. Yiu and K. P. Chow. A Secure Modified ID-Based Undeniable Signature Scheme based on Han et al.'s Scheme against Zhang et al.'s Attacks. Cryptology ePrint Archive, Report 2003/262, 2003. Available at http://eprint.iacr.org. 352

[10] Steven D. Galbraith, Keith Harrison, and David Soldera. Implementing the Tate Pairing. In Claus Fieker and David R. Kohel, editors, *Algorithmic Number Theory, 5th International Symposium, ANTS-V, Sydney, Australia, July 7-12, 2002, Proceedings* , volume 2369 of *Lecture Notes in Computer Science*, pages 324-337. Springer, 2002. 363

[11] Chandana Gamage, Jussipekka Leiwo, and Yuliang Zheng. Encrypted Message Authentication by Firewalls. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography: Second International Workshop on Practice and Theory in Public Key Cryptography, PKC 1999 Kamakura, Japan, March 1-3, 1999*, volume 1560 of *Lecture Notes in Computer Science*, pages 69-81. Springer-Verlag, Heidelberg, 1999. 352, 353

[12] Florian Hess. Efficient Identity Based Signature Schemes based on Pairings. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. Johns, Newfoundland, Canada, August*

15-16, 2002. Revised Papers, volume 2595 of *Lecture Notes in Computer Science*, pages 310-324. Springer-Verlag Heidelberg, February 2003. 353, 354

[13] Tetsuya Izu and Tsuyoshi Takagi. Efficient Computations of the Tate Pairing for the Large MOV Degrees. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference, Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 283-297. Springer-Verlag Heidelberg, 2003. 363

[14] Ik Rae Jeong, Hee Yun Jeong, Hyun Sook Rhee, Dong Hoon Lee, and Jong In Lim. Provably Secure Encrypt-then-Sign Composition in Hybrid Signcryption. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference, Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 16-34. Springer-Verlag Heidelberg, 2003. 353, 361

[15] Eunjeong Lee and YoungJu Choie. Implementation of Tate Pairing of Hyperelliptic Curves of Genus 2. In Jong In Lim and Dong Hoon Lee, editors, *Information Security and Cryptology - ICISC 2003, 6th Annual International Conference on Information Security and Cryptology, Seoul, Korea, Nov 2003, Proceedings* , Lecture Notes in Computer Science, Springer-Verlag. to appear. 363

[16] Benoît Libert and Jean-Jacques Quisquater. New Identity Based Signcryption Schemes from Pairings. In *IEEE Information Theory Workshop*, pages 155-158, 2003. Full version available at http://eprint.iacr.org. 352, 353, 354, 356, 361, 362, 366

[17] John Malone-Lee. Identity Based Signcryption. Cryptology ePrint Archive, Report 2002/098, 2002. Available at http://eprint.iacr.org. 353, 354, 356, 357

[18] Divya Nalla and K. C. Reddy. Signcryption scheme for Identity-Based Cryptosystems. Cryptology ePrint Archive, Report 2003/066, 2003. Available at http://eprint.iacr.org. 354

[19] David Pointcheval and Jacques Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology: The Journal of the International Association for Cryptologic Research*, 13(3):361-396, 2000. 368, 369

[20] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO 1984, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 47-53. Springer-Verlag, 1985. 352

[21] Jun-Bum Shin, Kwangsu Lee, and Kyungah Shim. New DSA-Verifiable Signcryption Schemes. In Pil Joong Lee and Chae Hoon Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference, Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 35-47. Springer-Verlag Heidelberg, 2003. 354

[22] Yuliang Zheng. Digital Signcryption or How to Achieve Cost (Signature & Encryption) << Cost(Signature) + Cost(Encryption). In Burton S. Kaliski Jr., editor, *Advances in Cryptology: Proceedings of CRYPTO 1997 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 165-179. Springer-Verlag, 1997. 352

Appendix

Proof of Theorem 1

Following the same idea as in [16], we assume that the distinguisher $\mathcal{C}$ receives a random instance $(P, aP, bP, cP, c^{-1}P, h)$ of the MDBDH problem and has to decide if $h = \hat{e}(P.P)^{abc}$. $\mathcal{C}$ will run $\mathcal{A}$ as a subroutine and act as $\mathcal{A}$'s challenger in the IND-IDSC-CCIA2 game. During the game, $\mathcal{A}$ will consult $\mathcal{C}$ for answers to the random oracles H_1, H_2 and H_3. Roughly speaking, these answers are randomly generated, but to maintain the consistency and to avoid collision, $\mathcal{C}$ keeps three lists L_1, L_2, L_3 respectively to store the answers used. The following assumptions are made.

(1) $\mathcal{A}$ will ask for $H_1(ID)$ before ID is used in any `Signcrypt`, `Unsigncrypt` and `Extract` queries.

(2) $\mathcal{A}$ will not ask for `Extract`(ID) again if the query `Extract`(ID) has been already issued before.

(3) Ciphertext returned from a `Signcrypt` request will not be used by $\mathcal{A}$ in an `Unsigncrypt` request.

$\mathcal{C}$ gives $\mathcal{A}$ the system parameters with $P_{pub} = cP$. Note that c is unknown to $\mathcal{C}$. This value simulates the master key value for the PKG in the game.

H_1 requests: When $\mathcal{A}$ asks queries on the hash values of identities, $\mathcal{C}$ checks the list L_1, If an entry for the query is found, the same answer will be given to $\mathcal{A}$; otherwise, a value d_i from $\mathbb{F}_q^*$ will be randomly generated and d_iP will be used as the answer, the query and the answer will then be stored in the list. Note that the associated private keys are d_icP and $d_ic^{-1}P$ which $\mathcal{C}$ knows how to compute.

The only exception is that $\mathcal{C}$ has to randomly choose one of the H_1 queries from $\mathcal{A}$, say the i-th query, and answers $H_1(ID_i) = bP$ for this query. Since bP is a value in a random instance of the MDBDH problem, it does not affect the randomness of the hash function H_1. Since both b, c and c^{-1} are unknown to $\mathcal{C}$, an `Extact` request on this identity will make $\mathcal{C}$ fails.

H_2, H_3 requests: When $\mathcal{A}$ asks queries on these hash values, $\mathcal{C}$ checks the corresponding list. If an entry for the query is found, the same answer will be given to $\mathcal{A}$; otherwise, a randomly generated value will be used as an answer to $\mathcal{A}$, the query and the answer will then be stored in the list.

`Signcrypt` requests: Let ID_A, ID_B be the identity of the sender and that of the recipient respectively and m be the plaintext used by $\mathcal{A}$ in a `Signcrypt` request. First we consider the simplest case that ID_A is not ID_i, then $\mathcal{C}$ can compute the private signcryption key S_{ID_A} correspondingly and the query can be answered by a call to `Signcrypt`(m, S_{ID_A}, Q_{ID_B}).

For the case $ID_A = ID_i$ and $ID_B \neq ID_i$, $\mathcal{C}$ answers `Signcrypt`(m, S_{ID_A}, Q_{ID_B}) query as follows. $\mathcal{C}$ randomly picks $r \in \mathbb{F}_q$ and $S \in \mathbb{G}_1^*$, computes $k_1 = \hat{e}(S, P_{pub})\hat{e}(Q_{ID_A}, P)^r$ and $\tau = \hat{e}(S, D_{ID_B})\hat{e}(Q_{ID_A}, Q_{ID_B})^r$ where D_{ID_B} is the private decryption key of ID_B. $\mathcal{C}$ finds $k_2 = H_2(\tau)$ by running the simulation for H_2 and computes $c = E_{k_2}(m)$. If there is a tuple (c, k_1, r') with $r' \neq r$ in L_3, $\mathcal{C}$ has to repeat the same process using another random pair (r, S) until the corresponding (c, k_1) does not appear in any tuple in L_3. This process repeats at most $2q_R$ times as L_3 contains at most $2q_R$ entries ($\mathcal{A}$ can issue q_R H_3 queries and q_R `Signcrypt` queries, while each `Signcrypt` query contains a single H_3 query). When an appropriate pair (r, S) is found, the ciphertext (c, r, S) appears to be valid from $\mathcal{A}$'s viewpoint. $\mathcal{C}$ has to compute 4 pairing operations for each iteration of the process.

The last case to consider is when both of ID_A and ID_B are the identity ID_i, $\mathcal{C}$ signcrypts m as follows. $\mathcal{C}$ chooses $r^* \in \mathbb{F}_q^*$ and $S^* \in \mathbb{G}_1$, computes $k_1^* = \hat{e}(S^*, P_{pub})\hat{e}(Q_{ID_A}, P)^{r^*}$ and randomly chooses $\tau^* \in_R \mathbb{G}_2$ and $k_2^* \in_R \{0,1\}^n$ such that no entry $(., k_2^*)$ is in L_2 and computes $c^* = E_{k_2^*}(m)$. He then checks if the list L_3 already contains an entry (c^*, k_1^*, r') such that $r' \neq r^*$. If not, $\mathcal{C}$ puts the tuple (c^*, k_1^*, r^*) into L_3 and (τ^*, k_2^*) into L_2. Otherwise, $\mathcal{C}$ chooses another random pair (r^*, S^*) and repeats the process as above until he finds a tuple (c^*, k_1^*, r^*) whose first two elements do not figure in an entry of L_3. Once an appropriate pair (r^*, S^*) is found, $\mathcal{C}$ gives the ciphertext $\sigma^* = (c^*, r^*, S^*)$ to $\mathcal{A}$. As $\mathcal{A}$ will not ask for the unsigncryption of σ^*, he will never see that σ^* is not a valid ciphertext of the plaintext m where $ID_A = ID_B = ID_i$ (since τ^* may not equal to $\hat{e}(S^*, D_{ID_B})\hat{e}(Q_{ID_A}, Q_{ID_B})^{r^*}$). $\mathcal{C}$ has to compute 2 pairing operations for each iteration of the process.

`Unsigncrypt` requests: First we consider the case when $\mathcal{A}$ observes a ciphertext $\sigma' = (c', r', S')$ from ID_A to ID_B where $ID_B = ID_i$. $\mathcal{C}$ always answers $\mathcal{A}$ that σ is invalid when $\mathcal{A}$ submits an `Unsigncrypt` request. For $\mathcal{C}$ to fail the simulation, $\mathcal{A}$ has made H_3 request with the tuple $(c', \hat{e}(S', P_{pub})\hat{e}(Q_{ID_A}, P)^{r'})$ before and $\mathcal{C}$ has answered r'. There is a probability of at most $1/2^k$ that $\mathcal{C}$ answered r' (and that σ' was actually valid from $\mathcal{A}$'s point of view) and L_3 actually contains a tuple $(c', \hat{e}(S', P_{pub})\hat{e}(Q_{ID_A}, P)^{r'}, r')$ (as $\mathcal{C}$ rejected a valid ciphertext).

For the case that $ID_B \neq ID_i$, $\mathcal{C}$ first computes $k_1' = \hat{e}(S', P_{pub})\hat{e}(Q_{ID_A}, P)^{r'}$. $\mathcal{C}$ rejects the ciphertext if the tuple (c', k_1', r') is not found in the list L_3; otherwise, he can recover r' and compute $\tau' = \hat{e}(S', D_{ID_B})\hat{e}(Q_{ID_A}, Q_{ID_B})^{r'}$. Note that the knowledge of D_{ID_B} can be simulated using the same technique in the simulation for the `Signcrypt` query. $\mathcal{C}$ then searches for a tuple $(\tau', \cdot)$ in list L_2. If no such tuple is found, $\mathcal{C}$ picks a random pair $(\tau, k_2') \in \mathbb{G}_2 \times \{0,1\}^n$ such that no tuples with k_2' already exists in L_2 and inserts (τ, k_2') in L_2. $\mathcal{C}$ can use the corresponding k_2' to find $m' = D_{k_2'}(c')$ and returns m'. The probability to reject at least 1 valid ciphertext is equal to $1 - ((2^k - 1)/2^k)^{q_U} = (q_U 2^{k(q_U-1)} - C_2^{q_U} 2^{k(q_U-2)} + \cdots)/2^{kq_U}$ which does not exceed $q_U/2^k$. For each unsigncryption request, $\mathcal{C}$ has to compute 4 pairing operations.

After the first stage, $\mathcal{A}$ picks a pair of identities on which he wishes to be challenged. Note that $\mathcal{C}$ fails if $\mathcal{A}$ has asked an `Extract` query on ID_i during the first stage. We know that the probability for $\mathcal{C}$ not to fail in this stage is $(\frac{q_H-1}{q_H})(\frac{q_H-2}{q_H-1})\cdots(\frac{q_H-q_E}{q_H-q_E+1}) = \frac{q_H-q_E}{q_H}$. Further, with a probability exactly $(\frac{q_H-q_E-1}{q_H-q_E})(\frac{1}{q_H-q_E-1}) = \frac{1}{q_H-q_E}$, $\mathcal{A}$ chooses to be challenged on the pair

(ID_j, ID_i) with $j \neq i$. Hence the probability that $\mathcal{A}$'s response is helpful to $\mathcal{C}$ is $\frac{1}{q_H}$. Note that if $\mathcal{A}$ has submitted an `Extract` query on ID_i, then $\mathcal{C}$ fails because he is unable to answer the question. On the other hand, if $\mathcal{A}$ does not choose (ID_j, ID_i) as target identities, $\mathcal{C}$ fails too.

Then $\mathcal{A}$ produces two plaintexts m_0 and m_1, $\mathcal{C}$ randomly picks a bit $b \in \{0, 1\}$ and signcrypts m_b. To do so, he sets $S' = aP$ and chooses $r' \in \mathbb{F}_q^*$. Suppose $ID_j = dP$, setting $S' = aP$ implies $(x - r')dc^{-1} = a$, i.e. $x = acd^{-1} + r'$. Since aP and cP belongs to a random instance of the MDBDH problem, x is random and this will not modify $\mathcal{A}$'s view. $\mathcal{C}$ computes $k_1' = \hat{e}(S', P_{pub})\hat{e}(Q_{ID_j}, P)^{r'} = \hat{e}(aP, cP)\hat{e}(Q_{ID_j}, P)^{r'}$, $\tau' = h\hat{e}(Q_{ID_j}, bP)^{r'}$ (where h is $\mathcal{C}$'s candidate for the MDBDH problem) to obtain $k_2' = H_2(\tau')$ (from the H_2 simulation algorithm) and $c_b = E_{k_2'}(m_b)$. He then verifies as above if L_3 already contains an entry (c_b, k_1', r'') such that $r'' \neq r'$. If not, he puts the tuple (c_b, k_1', r') into L_3. Otherwise, $\mathcal{C}$ picks another random r' and repeats the process until a tuple (c_b, k_1', r') whose first two elements do not appear in any entry of L_3 is found. After an appropriate r' is found, $\mathcal{C}$ sends the ciphertext $\sigma = (c_b, r', S')$ to $\mathcal{A}$.

$\mathcal{A}$ then performs another set of queries, $\mathcal{C}$ can handle these queries as in the first stage. At the end, $\mathcal{A}$ will produce a bit b' as $\sigma = \mathtt{Signcrypt}(m_{b'}, S_{ID_j}, Q_{ID_i})$ from $\mathcal{A}$'s viewpoint. If $b = b'$, $\mathcal{C}$ then answers 1 as the result to the MDBDH problem as he has produced a valid signcrypted message of m_b using the knowledge of h. Otherwise, $\mathcal{C}$ should answer 0.

Taking into account all the probabilities that $\mathcal{C}$ will not fail its simulation, the probability that $\mathcal{A}$ chooses to be challenged on the pair (ID_j, ID_i), and also the probability that $\mathcal{A}$ wins the IND-IDSC-CCIA2 game, the value of $Adv(\mathcal{C})$ is calculated as follows.

$$\begin{aligned} Adv(\mathcal{C}) &> (\frac{(\epsilon + 1)}{2}(1 - \frac{q_U}{2^k}) - 1/2)(\frac{1}{q_H}) \\ &= \frac{\epsilon(2^k - q_U) - q_U}{q_H 2^{(k+1)}} \end{aligned}$$

Regarding the time complexity, it can be verified by counting the number of pairing operations required to answer all queries. □

Proof of Theorem 2
We use the forking lemma [19] to prove the security of the scheme. To apply the forking lemma, we need to show how our scheme fits into the signature scheme described in [19], the simulation step in which the signature can be simulated without the secret signcryption key of the sender (and thus, also without the master secret), and how we can solve a difficult problem (MCBDH problem in our case) based on the forgery.

First, we observe that our scheme satisfies all the required properties of a generic signature scheme as described : during the signcryption of message m, the tuple (σ_1, h, σ_2) is produced which corresponds to the required three-phase honest-verifier zero-knowledge identification protocol, where $\sigma_1 = \hat{e}(xQ_{ID_A}, P)$ is the commitment of the prover (σ_1 can be considered to be chosen randomly from a large set since x is chosen randomly from $\mathbb{F}_q^*$ and $\mathbb{G}_2$ is a cyclic group of prime order q), $h = H_3(c, k_1)$ is the hash value depending on m and σ_1 (as c is a function of m) substituted for the verifier's challenge, and $\sigma_2 = S$ (which depends on x in σ_1, h and the signcryption key S_{ID_A}) is the response of the prover. As pointed out in [19], σ_1 can be omitted in the final signature produced in the scheme to optimize the size of signature since it can be correctly recovered during the verification process.

Next, we need to show a simulation step that provides a faithful simulation to the forger $\mathcal{A}$ and how to solve the MCBDH problem by interacting with $\mathcal{A}$. The distinguisher $\mathcal{C}$ receives a random instance $(P, aP, bP, cP, c^{-1}P)$ of the MCBDH problem and is required to compute $h = \hat{e}(P, P)^{abc}$. $\mathcal{C}$ will run $\mathcal{A}$ as a subroutine and act as $\mathcal{A}$'s challenger in the EUF-IDSC-CMIA2 game. $\mathcal{C}$ needs to maintain lists L_1, L_2, L_3 to keep track values reported for random oracle queries H_1, H_2, and H_3 to avoid collision and maintain consistency for answers to these hashing oracles. $\mathcal{C}$ publishes the system parameters and handles H_1, H_2, H_3, `Signcrypt` and `Unsigncrypt` requests in the same way as that in the proof of Theorem 1.

We calculate the probability of success of $\mathcal{C}$ as follows. For $\mathcal{C}$ to succeed, $\mathcal{A}$ did not ask an `Extract` query on ID_i. And the corresponding probability is at least $1/q_H$, as there are at most q_H entries in H_1. The probability of having a faithful simulation is at least $(1 - \frac{q_U}{2^k})(\frac{q_H - q_E}{q_H}) = \frac{(q_H - q_E)(2^k - q_U)}{q_H 2^k}$.

We follow the same idea used in [7] to coalesce the signing identity ID_i and the message m into a "generalized" forged message (ID_i, m) so as to hide the identity-based aspect of the EUF-IDSC-CMIA2 attacks, and simulate the setting of an identity-less adaptive-CMA existential forgery for which the forking lemma is proven.

It follows from the forking lemma that if $\mathcal{A}$ is a sufficiently efficient forger in the above interaction, then we can construct a Las Vegas machine $\mathcal{A}'$ that outputs two signed messages $((ID_i, m), r, S)$ and $((ID_i, m), r', S')$ with $r \neq r'$ and the same commitment x.

Finally, to solve the MCBDH problem given the machine $\mathcal{A}'$ derived from $\mathcal{A}$, we construct a machine $\mathcal{C}'$ as follows.

1. $\mathcal{C}'$ runs $\mathcal{A}'$ to obtain two distinct forgeries $((ID_i, m), r, S)$ and $((ID_i, m), r', S')$.
2. $\mathcal{C}'$ derives the value of $bc^{-1}P$ as $(r' - r)^{-1}(S - S')$ (since the value of x is the same in both forgeries).
3. $\mathcal{C}'$ derives the value of $\hat{e}(P, P)^{abc^{-1}}$ by $\hat{e}(aP, bc^{-1}P)$ (the role of c and c^{-1} are interchangeable).

Note that the machine $\mathcal{C}'$ is our reduction from the MCBDH problem. Based on the bound from the forking lemma [19] and the lemma on the relationship between given-identity attack and chosen-identity attack [8], if $\mathcal{A}$ succeeds in time $\leq t$ with probability $\epsilon \geq 10(q_R + 1)(q_S + q_R)q_H/(2^k - 1)$, then $\mathcal{C}'$ can solve the MCBDH problem in expected time $\leq 120686 q_R q_H 2^k t/\epsilon(2^k - 1)$. $\square$

Group Oriented Cryptosystems Based on Linear Access Structures[*]

Wen Ping Ma[1] and Moon Ho Lee[2]

[1] National Key Lab. of ISN
Xidian University, Xi'an 710071, P.R.China
wp_ma@hotmail.com
[2] Department of Information and Communication Engineering
Chonbuk National University, Korea
moonho@moak.chonbuk.ac.kr

Abstract. Group (or society) oriented cryptosystems are important in modern cryptography, they have wide application in areas such as group private communication and electronic commerce. Practical group oriented cryptosystems have been constructed using the techniques of secret sharing, but almost all such cryptosystems are based on threshold access structures, this is a great limitation to their application. In this paper, we consider group oriented cryptosystem based on vector space access strucures, a secure and practical group oriented cryptosystem based on vector space access structures is presented.

Keywords. Group Oriented Cryptosystem, ElGamal Cryptosystem, Secret Sharing, Vector Space Access Structure, Cryptography

1 Introduction

Group oriented cryptography was introduced by Y.Desmedt [1] in 1987, It has been further studied in [2, 3, 4]. When a group oriented cryptosystem is used in practice, anyone should be able to send encrypted messages to an organization (or a company, or a group) without knowing the private key for every member within the receiving organization, the receiving organization can set up its own security policy to determine who can read the encrypted messages it receives. In such systems, the sender cannot circumvent the security policy of the receiving organization, and he can send encrypted messages without knowing the policy, one basic property of the security policy is to require that the members of some specific subset of the organization to work together in order to be able to read the encrypted message.

Y. Desmedt et. al. proposed the first group oriented cryptosystem using Ping-pong protocol based on the method of Goldreich, Micali, and Wigderson [5], but

[*] This work was supported by the Institute of Information Technology Assessment ($IITA$), Chonbuk National University, Korea, and the National Science Foundation of China under the grant No. 60373104.

his solution is considered to be impractical. Y. Frankel [3] gave another solution using proxy but this system is lack of robustness. A practical system was presented by Yvo Desmedt and Yair Frankel [2] using threshold secret sharing, they also proposed the concept of threshold cryptography. A survey of the research on threshold cryptography can be found in [6]. In their threshold cryptosystem, a subset of the whole group can work together to decrypt an encrypted message only when the number of its members is not less than a certain number (the threshold). So this system is only useful when every member of the group has equally the same right, and no one has any advantages over the others. Unfortunately, in most cases, the members of a group do not have the same rights, for example, a manager of a company may have some advantages over the other members, hence, it is significant to generalize Yvo Desmedt and Yair Franke's group oriented cryptosystem to more general case.

In this paper, we will combine the ElGamal public key encryption scheme [7] and the techniques of linear secret sharing systems to present a group oriented cryptosystem based on vector space access structures, in the system, the members of each authorized subset of the group can work together to read any message encrypted by the group's public key.

2 Linear Secret Sharing Systems on Vector Space Access Structures

Linear secret sharing systems was first introduced by Brickell [8], more general vector space access structures were introduced by Simmons [9], his scheme was called geometric secret sharing system by Jackson and Martin [10], and monotone span program by Karchmer and Wigderson [11].

For completeness, we briefly introduce some basic facts about linear secret sharing systems. Let E be a vector space with finite dimension over finite field $GF(p)$ (where p is a prime or a prime power), the set of shareholders is denoted by $C = \{C_1, C_2, \cdots, C_n\}$. For any $C_i \in C \bigcup \{A\}$, where $A = C_0 \notin C$ is a special participant called *dealer* who holds the secret to be shared and distributes shares to each shareholder in C. For each $C_i \in \{C_0, C_1, \cdots, C_n\}$, let E_i be a vector space over $GF(p)$, and $\pi_i : E \to E_i$ be a linear subjection. Let us suppose that these linear mapping verify that, for any $X \subset C$,

$\bigcap_{C_i \in X} ker\pi_i \subset ker\pi_0$ or $\bigcap_{C_i \in X} ker\pi_i + ker\pi_0 = E$.

This family of vector spaces and linear surjective mappings determines the access structure

$$\Gamma = \{X \subset C : \cap_{C_i \in X} ker\pi_i \subset ker\pi_0\}.$$

The distribution algorithm for secret space E_0 and access structure Γ is as follows. For secret $s \in E_0$, the dealer selects at random a vector $v \in E$ satisfying $\pi_0(v) = s$. He then secretly sends $s_i = \pi_i(s) \in E_i$ to each shareholder $C_i \in C, i = 1, 2, \cdots, n$.

Next, we describe the reconstruction algorithm in detail as it plays an important role in our new group oriented cryptosystem.

Let $dimE_i = k_i$, where $i = 0, 1, , 2, \cdots, n$, $dimE = k$. It is widely known that there is a bijection between the linear maps from E_i to E and the $k_i \times k$ matrices over $GF(p)$. Suppose B_i is the matrix corresponding to $\pi_i, i = 0, 1, 2, \cdots, n$. Denoting $s = (a_1, a_2, \cdots, a_{k_0}), v = (b_1, b_2, \cdots, b_k)$, and $B_0 = (e_{ij}), 1 \le i \le k_0, 1 \le j \le k$, then $\pi_0(v) = s$ corresponds to $B_0 v^T = s$.

Let $\{C_{i_1}, C_{i_2}, \cdots, C_{i_l}\}$ be an authorized subset of C, for simplicity, we use $\{i_1, i_2, \cdots, i_l\}$ instead of $\{C_{i_1}, C_{i_2}, \cdots, C_{i_l}\}$.

When to reconstruct the shared secret, the members of $\{i_1, i_2, \cdots, i_l\}$ use a $(k_{i_1} + k_{i_2} + \cdots + k_{i_l}) \times k$ matrix

$$B = \begin{pmatrix} B_{i_1} \\ B_{i_2} \\ \vdots \\ B_{i_l} \end{pmatrix}$$

composed of matrices $B_{i_1}, B_{i_2}, \cdots, B_{i_l}$. They use their shares $s_{i_1}, s_{i_2}, \cdots, s_{i_l}$ to form a $k_{i_1} + k_{i_2} + \cdots + k_{i_l}$ dimension row vector $y = (s_{i_1}, s_{i_2}, \cdots, s_{i_l})$. Now they can find a solution v of the equation $B \times v^T = y^T$ and compute the secret $s = \pi_0(v)$. The above procedure can be accomplished using the following three steps :

1. Find two nonsingular matrices P, C such that

$$PBC = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots\dots & \dots & \dots & \\ 0 & 0 & \dots & 1 & \dots & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix}$$

   where P is of order $k_{i_1} + k_{i_2} + \cdots + k_{i_l} = k'$, and C of order k.
2. Find a solution v_1 of equation $(PBC)v^T = Py^T$, and then compute $v^T = Cv_1^T$ (since $PBC(C^{-1}v^T) = Py^T$) .
   Let $P = (p_{ij}), 1 \le i, j \leqq k', C = (c_{ij}), 1 \leqq i, j \leqq k, r = Rank(B) \leqq min\{k, k'\}$. Denote $y = (y_1, y_2, \cdots, y_{k'})$, then

$$\begin{aligned} Py^T &= (\Sigma_{j=1}^{k'} p_{1j} y_j, \Sigma_{j=1}^{k'} p_{2j} y_j, \cdots, \Sigma_{j=1}^{k'} p_{k' j} y_j) \\ &= (\Sigma_{j=1}^{k'} p_{1j} y_j, \Sigma_{j=1}^{k'} p_{2j} y_j, \cdots, \Sigma_{j=1}^{k'} p_{rj} y_j, 0, \cdots, 0) \end{aligned}$$

   A solution of equation $(PBC)v_1^T = Py^T$ is a k dimensional vector

$$v_1 = (\Sigma_{j=1}^{k'} (p_{1j} y_j), \Sigma_{j=1}^{k'} (p_{2j} y_j), \cdots, \Sigma_{j=1}^{k'} (p_{rj} y_j), 0, \cdots, 0).$$

   Denote it by

$$v_1 = (x_1, x_2, \cdots, x_r, 0, \cdots, 0)$$

   and $v = (\Sigma_{j=1}^{r} c_{1j} x_j, \Sigma_{j=1}^{r} c_{2j} x_j, \cdots, \Sigma_{j=1}^{r} c_{kj} x_j)$ satisfies $v^T = Cv_1^T$.

3. Compute

$$s = \pi_0(v) = (\Sigma_{j=1}^{k} e_{1j} z_j, \Sigma_{j=1}^{k} e_{2j} z_j, \ldots, \Sigma_{j=1}^{k} e_{k_0 j} z_j) = (a_1, a_2, \cdots, a_{k_0}),$$

where $z_m = \Sigma_{j=1}^{r} c_{mj} x_j, m = 1, 2, \cdots, k$.

3 A Variation of ElGamal Encryption Scheme

The ElGamal encryption scheme [7] is based on the discrete-log problem. Here we give a variation of the ElGamal encryption scheme, it will be used in the linear secret sharing systems described above.

Let $GF(q)$ be a finite field, where q is a prime, let p a prime such that $p|(q-1)$, and g an element of $GF(q)$ with order p. The secret key is $s = (a_1, a_2, \cdots, a_{k_0}) \in E_{k_0}$, and the corresponding public key is $(g^{a_1}, g^{a_2}, \cdots, g^{a_{k_0}})$. To encrypt a message M the sender first represents it as a vector $M = (M_1, M_2, \cdots, M_{k_0}) \in E_0$, where $M_i \in GF(q), i = 1, 2, \cdots, k_0$. Then the sender randomly selects $k \in GF(p)$, computes and sends ciphertext

$$C = (g^k, M_1 g^{a_1 k}, M_2 g^{a_2 k}, \cdots, M_{k_0} g^{a_{k_0} k}) mod\ q = (c_0, c_1, \cdots, c_{k_0}).$$

For decryption, the recipient computes

$$M_i' = c_i c_0^{-a_i} (mod\ q), i = 1, 2, \cdots, k_0,$$

and the plain-text is $M' = (M_1', M_2', \cdots, M_{k_0}')$.

4 A Group Oriented Cryptosystem Based on Linear Secret Sharing

The system is divided into three phases, i.e. key generation and distribution, encryption, and decryption.

4.1 Key Generation and Distribution

Let $C = (C_1, C_2, \cdots, C_n)$ be the group, $A = C_0 \notin C$ be a trusted authority who is responsible for key generation and distribution of shares of the group secret key to the group members. The notations $E, E_i, B, B_i, k, k_i, i = 1, 2, \cdots, n$, and Γ (the vector space access structure on C) are the same as in section 2. The parameters p and q are the same as in section 3.

The trusted authority A randomly selects the group's private key, $s = (a_1, a_2, \cdots, a_{k_0}) \in E_{k_0}$ computes and publishes the corresponding public key $(g^{a_1}, g^{a_2}, \cdots, g^{a_{k_0}})$. To distribute the shares of the group's private key s, A uses the linear secret sharing system described in section 2, he selects at random a vector $v \in E_0$ such that $\pi_0(v) = s$, computes and sends secretly $s_i = \pi_i(v)$ to the group member $C_i, i = 1, 2, \cdots, n$. s_i is C_i's secret share (or private key).

4.2 Encryption

To encrypt a message M, the sender first represents it as a vector in E_0, i.e.

$$M = (M_1, M_2, \cdots, M_{k_0}), M_i \in GF(q), i = 1, 2, \cdots, k_0.$$

Next, the sender selects at random a $k \in GF(p)$, computes and sends

$$D = (g^k, M_1 g^{a_1 k}, M_2 g^{a_2 k}, \cdots, M_{k_0} g^{a_{k_0} k})$$

to the group.

4.3 Decryption

When receiving the encrypted message $D = \{c, c_1, c_2, \cdots, c_{k_0}\}$, the members of each authorized subset of the group C can work together to decrypt it. Suppose $H = \{C_{i_1}, C_{i_2}, \cdots, C_{i_l}\}$ is an authorized subset of C, and the members of H cooperate to decrypt cipher-text D. Let the secret share of C_{i_j} is $s_{i_j} = (a_1^{i_j}, a_2^{i_j}, \cdots, a_{k_{i_j}}^{i_j})$.

C_{i_j} computes $(c^{a_1^{i_j}}, c^{a_2^{i_j}}, \cdots, c^{a_{k_{i_j}}^{i_j}})$ $(mod\ q)$ and send it to a designated individual (Dealer) who is responsible for compute the plain-text . The designated individual carries out the following steps :

1. Uses $(c^{a_1^{i_j}}, c^{a_2^{i_j}}, \cdots, c^{a_{k_{i_j}}^{i_j}})$ $(mod\ q), j = 1, 2, \cdots, l$ to form a row vector
$$(c^{y_1}, c^{y_2}, \cdots, c^{y_{k'}}) = (c^{a_1^{i_1}}, c^{a_2^{i_1}}, c^{a_{k_{i_1}}^{i_1}}, \cdots, c^{a_1^{i_l}}, c^{a_2^{i_l}}, \cdots, c^{a_{k_{i_l}}^{i_l}})$$
where $k' = k_{i_1} + k_{i_2} + \cdots + k_{i_l}$. Similar to the reconstruction algorithm of linear secret sharing system, he constructs a matrix B from the maps $\pi_{i_1}, \pi_{i_2}, \cdots, \pi_{i_l}$, finds the nonsingular matrices P, C, and then computes $(c^{x_1}, c^{x_2}, \cdots, c^{x_r}, 1, \cdots, 1)$,
where $c^{x_j} = \Pi_{m=1}^{k'} (c^{y_m})^{p_{jm}} (mod\ q), j = 1, 2, \cdots, r = Rank(B)$.
2. Computes $(c^{z_1}, c^{z_2}, \cdots, c^{z_k})$,using matrix $C = (c_{ij})$, where
$$c^{z_m} = \Pi_{j=1}^{r} (c^{x_j})^{c_{mj}} (mod\ q), m = 1, 2, \cdots, k.$$
3. Computes $(c^{a_1}, c^{a_2}, \cdots, c^{a_{k_0}})$, using linear map π_0 (matrix $B_0 = (e_{ij})$), where
$$c^{a_m} = \Pi_{j=1}^{k} (c^{z_j})^{e_{mj}} (mod\ q), m = 1, 2, \cdots, k_0.$$
4. Computes plain-text
$$M = (c_1 c^{-a_1}, c_2 c^{-a_2}, \cdots, c_{k_0} c^{-a_{k_0}})(mod\ q)$$
and securely broadcasts it to all members of H.

5 Security and Performance Analysis

Our group oriented cryptosystem is a combination of linear secret sharing systems with vector space access structures and the ElGamal encryption scheme. The secret sharing systems with vector space access structures we used are unconditional secure. The ElGamal encryption scheme is computational secure. So our newly proposed group oriented cryptosystem is computational under the assumption that compute discrete logarithm in finite field $GF(q)$ is infeasible.

In decryption phase, the designated individual can compute the plain text if all members of an authorized subset of the group send correct information $((c^{a_1^{i_j}}, c^{a_2^{i_j}}, \cdots, c^{a_{k_{i_j}}^{i_j}})$ in the above) to him. However, the designated individual cannot get the secret share (private key) of any group member's on condition that compute discrete logarithm in finite field $GF(q)$ is infeasible.

6 Conclusion

We have presented a group oriented cryptosystem using generalized secret sharing on vector space access structures. The system is based on a simple variation of the ElGamal public encryption scheme. It is a generalization of Yvo Desmedt and Yair Frankel's threshold cryptosystem. The efficiency of the system can be improved by optimizing the parameters of the underline linear secret sharing scheme.

References

[1] Y. Desmedt, Society and group oriented cryptography : a new concept, Advance in Cryptology, Proc. of Crypto '87, Lecture Notes in Computer Science, Lncs 293, pp. 120-127, Springer-Verlag, 1988. 370

[2] Yvo Desmedt, Yair Frankel, Threshold Cryptosystems, Advance in Cryptology – Crypto' 89, Lecture Notes in Computer Science, pp.305-315, Springer-Verlag, 1989. 370, 371

[3] Y.Frankel, A practical protocol for large group oriented networks , Advance in Cryptology, Proc.of Eurocrypt' 89, Lecture Notes in Computer Science, pp.56-61, Springer-Verlag, 1989. 370, 371

[4] T. Wang, Cryptosystem for group oriented cryptography, in Advances in Cryptology, proceedings of Eurocrypt ' 90, Lecture Notes in Computer Science, pp.352-360, pringer-Verlag, 1990. 370

[5] O. Goldreich, S. Micali, and A. Wigderson, How to play an mental game, in Proceedings of the Nineteenth ACM Symp, Theory of Computing, pp. 218-229, May 25-27, 1987. 370

[6] Yvo Desmedt, Some Recent research aspects of threshold cryptography, In E. Okamoto, G.Davida and M. Mambo(eds), Information Security, Lecture Notes in Computer Science, pp 158-173, Springer-Verlag, 1997. 371

[7] T.ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Info. Theory, Vol.31, No.4, 1985, pp.469-472. 371, 373

[8] E. F. Brickell, Some ideal secret sharing schemes, The Journal of Combinatorial Mathematics and Combinatorial Computing, Vol.6, pp.105-113, 1989. 371

[9] G. J. Simmons, How to (really) Share a secret, in Proceedings of Crypto '88, Lecture Notes in computer science, Vol.403, pp.390-448, Springer-Verger, 1990. 371

[10] W. Jackson and K. Martin, Geometric Secret sharing schemes and their duals, Design, Codes, and Cryptography, No.4, pp.83-95, 1994. 371

[11] M. Karchmer, A. Wigderson, On Span Programs, in Proc.of the 8th Annual IEEE Symposium on Structure in Complexity, pp.102-111, 1993. 371

A New Algorithm for Searching a Consistent Set of Shares in a Threshold Scheme with Cheaters

Raylin Tso[1], Ying Miao[2], and Eiji Okamoto[3]

[1] Risk Engineering Major
Graduate School of Systems and Information Engineering
University of Tsukuba, Tsukuba 305-8573, Japan
raylin@sk.tsukuba.ac.jp
[2] Institute of Policy and Planning Sciences
University of Tsukuba, Tsukuba 305-8573, Japan
miao@sk.tsukuba.ac.jp
[3] Institute of Information Sciences and Electronics
University of Tsukuba, Tsukuba 305-8573, Japan
okamoto@is.tsukuba.ac.jp

Abstract. In a (k, n) Shamir's threshold scheme, if one or more of the n shares are fake, then the secret may not be reconstructed correctly by some sets of k shares. Supposing that at most t of the n shares are fake, Rees et al. (1999) described two algorithms to determine consistent sets of shares so that the secret can be reconstructed correctly from k shares in any of these consistent sets. In their algorithms, no honest participant can be absent and at least $n-t$ shares should be pooled during the secret reconstruction phase. In this paper, we propose a modified algorithm for this problem so that the number of participants taking part in the secret reconstruction can be reduced to $k+2t$ and the shares need to be pooled can be reduced to, in the best case, $k+t$, and less than or equal to $k+2t$ in the others. Its efficiency is also investigated.

1 Introduction

Threshold schemes for secret sharing were introduced by Blakley [1] and Shamir [7] independently in 1979. Informally, a (k, n) *threshold scheme* is a method of breaking a secret K up into n different shares $S_1, S_2, \ldots, S_n$ such that

1. with the knowledge of any k or more shares ($k \leq n$), the secret K can be easily derived; and
2. with the knowledge of any $k-1$ or fewer shares, it is impossible to derive the secret K.

Shamir's threshold scheme is based on the Lagrange interpolating polynomial where the constant term is the secret K. In this paper we will adopt Shamir's scheme for the convenience of description.

The problem of detecting cheaters in threshold schemes was investigated by McEliece and Sarwate [4], and others (see, for example, [2], [3], [5], [8]). Under the assumption that at most t of the n shares are fake, McEliece and Sarwate [4] showed that if $k + 2t$ shares are pooled together, then there exists a unique polynomial of degree at most $k - 1$ so that at least $k + t$ of the genuine shares lie on it. This unique polynomial, which is used to compute the secret, can be efficiently determined by both the Berlekamp-Massey algorithm or the Euclidean algorithm for Reed-Solomon codes.

Rees et al. [6] considered the problem of determining consistent sets of shares in a threshold scheme with cheaters. A set of shares in a (k, n) threshold scheme is called *consistent* if every k-subset of the shares derives the same secret. Rees et al. proposed a deterministic algorithm, based on a particular class of t-coverings with index 1, to reconstruct the secret of a (k, n) Shamir's threshold scheme when some shares are fake. They also considered a randomized algorithm for the same problem, and compared it with the above mentioned deterministic algorithm. Their underlying idea is to find a suitable set system $(S, \mathcal{T})$, where S is the set of all n shares and $\mathcal{T}$ is a collection of k-subsets of S, so that for any t-subset S_t of shares (thus the subset of all fake shares, if we assume that at most t of the n shares are fake), there is at least one $T \in \mathcal{T}$ such that T doesn't contain any share in this t-subset S_t. Then the k-subset of $\mathcal{T}$ containing no fake shares can be used to derive the secret correctly.

Unfortunately, one drawback of Rees et al. 's algorithms is that they sacrifice the property of *threshold*. That is, no honest participant can be absent if they decide to reconstruct the secret while in a conventional (k, n) threshold scheme, only k of the n participants are needed. As a consequence, Rees et al. 's algorithms are not so practical in a (k, n) threshold scheme where k and t are small but n is relatively large, or gathering all the participants together is impossible. In this paper, we propose a modified algorithm for this problem so that the number of participants taking part in the secret reconstruction can be reduced to $k + 2t$ if at most t of the n shares are fake. In our algorithm, the shares need to be pooled can be reduced to, in the best case, $k + t$, and less than or equal to $k + 2t$ in the others. Some efficiency will be sacrificed in our algorithm in order to allow any $k + 2t$ of the n participants to achieve the end of determining a consistent set of shares in a threshold scheme with at most t cheaters. This is verified by the numerical results of our computer experiments. The expected number of shares used to reconstruct the secret will also be compared with those of Rees et al. 's algorithms [6] and $k + 2t$, the number of shares needed to reconstruct the secret by McEliece and Sarwate's method [4], in some special cases.

2 Rees et al. 's Two Algorithms

Suppose that the (k, n) Shamir's threshold scheme is implemented in $GF(q)$. Let

$$S = \{(x_i, y_i) : \ 1 \le i \le n\} \subseteq (GF(q) \setminus \{0\}) \times GF(q)$$

be the set of n shares, and assume that at most t of the n shares are fake. That is, there exists a polynomial $P_0(x) \in GF(q)[x]$ of degree at most $k - 1$ such that $y_i = P_0(x_i)$ for at least $n-t$ of the n shares. The secret, which can be reconstructed from any k genuine shares, is the value $P_0(0)$.

Denote the subset of *good shares* by $G = \{i : \ y_i = P_0(x_i), 1 \le i \le n\}$, and the subset of *bad shares* by $B = \{1, 2, \ldots, n\} \setminus G$. Then $|G| \ge n-t$ and $|B| \le t$.

For any $T \subseteq \{1, 2, \ldots, n\}$ such that $|T| = k$, there is a unique polynomial P_T of degree at most $k-1$ such that $P_T(x_i) = y_i$ for all $i \in T$, which can easily be computed by Lagrange interpolation

$$P_T(x) = \sum\nolimits_{i \in T} y_i \prod\nolimits_{j \in T \setminus \{i\}} \frac{x - x_j}{x_i - x_j}.$$

The following two facts are obvious:

1. If $T \subseteq G$, then $P_T = P_0$.
2. If $T \cap B \ne \emptyset$, then $P_T \ne P_0$.

Define $C_T = \{i : \ P_T(x_i) = y_i, 1 \le i \le n\}$. Then $|C_T| \ge n-t$ if $T \subseteq G$, and $|C_T| \le k+t-1$ if $T \cap B \ne \emptyset$. If $n-t \le k+t-1$, then there could exist a polynomial $P_T \ne P_0$ of degree at most $k-1$ such that at least $n-t$ shares lie on P_T. Therefore, Rees et al. [6] required that the inequality $n \ge k+2t$ always holds.

Let v, k, λ and t be positive integers such that $v \ge k \ge t$. A t-(v,k,λ) *covering* is a pair $(\mathcal{V}, \mathcal{B})$, where $\mathcal{V}$ is a v-set of elements, called *points*, and $\mathcal{B}$ is a collection of k-subsets of $\mathcal{V}$, called *blocks*, such that every t-subset of points occurs in at least λ blocks of $\mathcal{B}$. The *covering number* $C_\lambda(v,k,t)$ is the minimum number of blocks in any t-(v,k,λ) covering. A t-(v,k,λ) covering $(\mathcal{V}, \mathcal{B})$ is *optimal* if $|\mathcal{B}| = C_\lambda(v,k,t)$.

Let $\mathcal{T}$ be a collection of k-subsets of $\{1, 2, \ldots, n\}$ such that its complement $\{\{1, 2, \ldots, n\} \setminus T : T \in \mathcal{T}\}$ forms the collection of blocks of a t-$(n, n-k, 1)$ covering. Then Rees et al. [6] claimed that the following deterministic algorithm can compute the polynomial P_0.

Algorithm 1
Input $\mathcal{T}, S, n, k, t$.

```
    For each $T \in \mathcal{T}$, perform the following steps:

1. compute $P_T$
2. compute $C_T$
3. **if** $|C_T| \ge n-t$ **then** $P_0 = P_T$ and QUIT
```

Rees et al. [6] also provided a randomized algorithm to compute P_0.

Algorithm 2
Input S, n, k, t.

```
    REPEAT the following steps:

1. let $T$ be a random $k$-subset of $\{1, 2, \ldots, n\}$
2. compute $P_T$
```

```
3. compute $C_T$
4. **if** $|C_T| \geq n - t$ **then**
   $P_0 = P_T$ and QUIT
   **else**
   proceed to the next iteration of the REPEAT loop
```

Algorithm 2 is not as efficient as Algorithm 1 because $C_1(n, n-k, t)$ provides an upper bound on the number of iterations required by Algorithm 1.

3 The New Practical Algorithm

As mentioned in the first section, a drawback of Rees et al. 's algorithms is that all the honest participants should be gathered together in order to make the secret reconstruction successful in any situation. On the other hand, we also note that if we apply Algorithm 1 of Rees et al. 's method to a (k, n_s) multiple threshold scheme of s multiplicity with at most t cheaters, or apply to s different (k, n_s) threshold schemes with at most t cheaters each and the same threshold value k, but varying in the number of participants n_s, then we need to construct and store s different (optimal) t-$(n_s, n_s - k, 1)$ coverings at first. But some of the required t-$(n_s, n_s - k, 1)$ coverings may be unknown or hard to be constructed. In addition, storing all the data of these coverings may consume a lot of capacity or cost expensively. In this section, we modify Rees et al. 's Algorithm 1 so that the new algorithm can overcome all the disadvantages described above.

According to Rees et al. [6], as was described in Section 2, $|C_T| \leq k + t - 1$ if $T \cap B \neq \emptyset$. Therefore, if $|C_T| \geq k + t$, then it must be the case that $T \cap B = \emptyset$. On the other hand, if we define $NC_T = \{i : P_T(x_i) \neq y_i, 1 \leq i \leq n\}$, then $|NC_T| \leq t$ if $T \cap B = \emptyset$. As a consequence, the following two facts become obvious:

1. If $|C_T| \geq k + t$, then $P_T = P_0$.
2. If $|NC_T| \geq t + 1$, then $P_T \neq P_0$.

Let $\mathcal{R}$ be a $(k + 2t)$-subset of $\{1, 2, \ldots, n\}$, $S_{\mathcal{R}} = \{(x_i, y_i) : i \in \mathcal{R}\} \subseteq S$ and $\mathcal{T}$ be a collection of k-subsets of $\mathcal{R}$ such that its complement $\{\mathcal{R} \setminus T : T \in \mathcal{T}\}$ is the collection of blocks of a t-$(k + 2t, 2t, 1)$ covering. Then the following deterministic algorithm will compute the polynomial P_0 for any $k + 2t$ of the n participants. In this algorithm, for convenience, we denote $\mathcal{R} \setminus T = \{r_{i_1}, \ldots, r_{i_{2t}}\}$ for each $T \in \mathcal{T}$, where the subscripts are ordered randomly.

Algorithm 3
Input $\mathcal{R}, \mathcal{T}, S_{\mathcal{R}}, k, t$.

For each $T \in \mathcal{T}$, perform the following steps:

```
1. compute $P_T$
2. $|NC_T| = 0$
3. $|C_T| = k$
```

```
4. for $j = 1$ to $2t$ do
5.     if $y_{r_{i_j}} = P_T(x_{r_{i_j}})$, then $|C_T| + +$
6.     else $|NC_T| + +$
7.     if $|C_T| \geq k + t$, then $P_0 = P_T$ and QUIT
8.     else if $|NC_T| \geq t + 1$ then BREAK
```

The advantage of adding the computation of NC_T is that if $P_T \neq P_0$, then the *for* loop can be broken out at the point $|NC_T| = t+1$ and the computation for the next T can be proceeded, while in Algorithms 1 and 2, $P_T \neq P_0$ will lead the algorithms to run without stop until all the shares have been pooled and result in $|C_T| < k + t$. Another advantage is that one and only one of the two conditions in steps 7 and 8 is satisfied in each loop, and there is at least one k-subset $T \in \mathcal{T}$ such that $|C_T| \geq k + t$, so adding the computation of NC_T makes any $k + 2t$ of the n participants possible to reconstruct the secret.

For each of the three algorithms, the best case occurs when the secret is successfully reconstructed at the first try and the algorithm is ended in $P_T = P_0$ with all the pooled shares belonging to C_T. In this case, only $k + t$ shares are needed in Algorithm 3 while $n - t$ shares are needed in Algorithms 1 and 2.

4 Performance Evaluation of the Algorithms

In this section, we assume that there are exactly t bad shares. We evaluate these three algorithms for their efficiencies and the total shares need to be pooled.

Let β_r be the expected number of iterations of the randomized Algorithm 2. Then it is not difficult to know (see [6]) that

$$\beta_r = \frac{\binom{n}{k}}{\binom{n-t}{k}}.$$

However, the expected numbers of iterations of Algorithm 1 and Algorithm 3 depend on the specific set systems $\mathcal{T}$ they used. Since there is no uniform description of "good" t-$(n, k, 1)$ coverings, we can not expect to find a unified theoretical method for the computation of the expected numbers of iterations of Algorithm 1 and Algorithm 3. Instead, we set up a computer experiment to compare the expected number β_{d_3} of iterations of the deterministic Algorithm 3 with β_{d_1} and β_r, those of Algorithms 1 and 2, and to compare the average number s_{d_3} of shares needed in Algorithm 3 to reconstruct the secret correctly with s_{d_1}, that needed in Algorithm 1, and the value $k + 2t$, the number of shares needed in McEliece and Sarwate's method [4].

To avoid any source of bias, in our experiment, we neither construct any specific polynomial $P_0(x) \in GF(p)[x]$ nor construct all the genuine shares (x_i, y_i) for $i \in G$ and all the fake shares (x_j, y_j) for $j \in B$. This causes no problem for the computation of the expected number of iterations of each algorithm. But when we compute the expected number of shares needed in Algorithm 3, for some k-subset T of $\mathcal{R}$ with $P_T \neq P_0$, we will not be able to verify whether a share belongs to C_T or NC_T without the knowledge of the polynomial P_T and all the shares belonging to $S_{\mathcal{R}}$. Therefore, we make an assumption that if $P_T \neq P_0$, then in Algorithm 3, the good shares not belonging to T will be counted in NC_T. This assumption is reasonable if the Shamir's threshold scheme is implemented in $GF(q)$ with $q >> max\{k, 2t\}$. This will be explained right after Lemma 1. We also assume that the bad shares not belonging to T will be counted in C_T in Algorithm 3 if $P_T \neq P_0$, since any cheater wants to mislead other participants to some incorrect secret and thus he/she will try his/her best to make his/her bad share belong to C_T, which, as a by-product, forces more shares to be pooled. This assumption also results in an overestimation of the number of shares needed in Algorithm 3 if the fake shares are caused by communication noise instead of the existence of cheaters.

Lemma 1. Let $P_0(x) \in GF(q)[x]$ and $P_T(x) \in GF(q)[x]$ be two different polynomials of degree at most $k-1$. For any randomly chosen u-subset of points from $P_0 \in GF(q)[x]$, if $q >> max\{k, u\}$, then the probability that at least one of these u points lies on P_T is nearly equal to 0.

Proof. The number of points which lie on both P_0 and P_T is at most $k-1$ and there are totally q points on $P_0(x) \in GF(q)[x]$ and on $P_T(x) \in GF(q)[x]$, respectively. Let Q_0 and Q_T denote the sets of these q points respectively. Then, for any randomly chosen u-subset $U \subseteq Q_0$, the probability that $U \cap Q_T \neq \emptyset$ will be

$$\begin{aligned} &Prob(U \cap Q_T \neq \emptyset) = 1 - Prob(U \cap Q_T = \emptyset) \\ &\leq 1 - \frac{\binom{q-k+1}{u}}{\binom{q}{u}} = 1 - \frac{(q-k+1) \times \cdots \times (q-k+2-u)}{q \times \cdots \times (q-u+1)} \\ &\approx 0, \end{aligned}$$

where the inequality comes from $|Q_0 \cap Q_T| \leq k-1$. □

In our experiment for Algorithm 3, $|(\mathcal{R} \setminus T) \cap G| \leq |\mathcal{R} \setminus T| = 2t$ holds for any $T \in \mathcal{T}$, and if $|T \cap B| = k_1 > 0$, then $|T \cap G| = k - k_1$. Meanwhile, $|\{(x_i, y_i) : y_i = P_0(x_i), x_i \neq 0\}| = q-1$, and at most $k-1$ of these $q-1$ points may lie on P_T because the degrees of P_0 and P_T are at most $k-1$ and $P_0 \neq P_T$. Since $k-k_1$ of these common points have been pooled, by Lemma 1, the probability that at least one of the remaining (at most $k_1 - 1$) good shares lying on P_T can be included in the $2t$-set $\mathcal{R} \setminus T$ is nearly equal to 0, and therefore it can be ignored.

Table 1 reports the numerical results of our experiment. The values of β_{d_1} and β_r are taken from Table 4 of Rees et al. [6]. The twenty-seven t-$(n, n-k, 1)$ coverings with different parameters n and k used in Algorithm 1 are listed below: $k = 3$ and $t = 2$: the following blocks form a 2-$(n, n-3, 1)$ covering over $\{1, 2, \ldots, n\}$ for any $n \geq 9$:

$$\{1, 2, \ldots, n\} \setminus \{1, 2, 3\}, \quad \{1, 2, \ldots, n\} \setminus \{4, 5, 6\}, \quad \{1, 2, \ldots, n\} \setminus \{7, 8, 9\}.$$

$k = 4$ and $t = 2$: the following blocks form a 2-$(n, n-4, 1)$ covering over $\{1, 2, \ldots, n\}$ for any $n \geq 12$:

$$\{1,2,\ldots,n\} \setminus \{1,2,3,4\}, \{1,2,\ldots,n\} \setminus \{5,6,7,8\},$$
$$\{1,2,\ldots,n\} \setminus \{9,10,11,12\}.$$

$k = 4$ and $t = 3$: the following blocks form a 3-$(n, n-4, 1)$ covering over $\{1, 2, \ldots, n\}$ for any $n \geq 16$:

$$\{1,2,\ldots,n\} \setminus \{1,2,3,4\}, \quad \{1,2,\ldots,n\} \setminus \{5,6,7,8\},$$
$$\{1,2,\ldots,n\} \setminus \{9,10,11,12\}, \{1,2,\ldots,n\} \setminus \{13,14,15,16\}.$$

For the set systems used in Algorithm 3, we use three coverings from the web page [9]: a 2-$(7, 4, 1)$ covering with $k = 3$ and $t = 2$, a 2-$(8, 4, 1)$ covering with $k = 4$ and $t = 2$, and a 3-$(10, 6, 1)$ covering with $k = 4$ and $t = 3$, which are listed below.
$\mathcal{R} = \{1, 2, 3, 4, 5, 6, 7\}$: the following blocks form a 2-$(7, 4, 1)$ covering over $\mathcal{R}$:

$$\{1,2,3,4\}, \{1,4,5,6\}, \{1,5,6,7\},$$
$$\{2,3,4,7\}, \{2,3,5,6\}.$$

$\mathcal{R} = \{1, 2, 3, 4, 5, 6, 7, 8\}$: the following blocks form a 2-$(8, 4, 1)$ covering over $\mathcal{R}$:

$$\{1,2,3,4\}, \{1,4,5,8\}, \{1,5,6,7\},$$
$$\{2,3,5,8\}, \{2,3,6,7\}, \{4,6,7,8\}.$$

$\mathcal{R} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$: the following blocks form a 3-$(10, 6, 1)$ covering over $\mathcal{R}$:

$$\{1,2,3,4,5,7\}, \{1,2,3,5,9,10\}, \{1,2,3,4,6,10\},$$
$$\{1,2,4,8,9,10\}, \{1,3,7,8,9,10\}, \{1,5,6,7,8,9\},$$
$$\{2,3,4,5,6,8\}, \{2,6,7,8,9,10\}, \{3,4,5,6,7,9\},$$
$$\{4,5,6,7,8,10\}.$$

In the experiment, the bad shares are selected randomly in each of the cases. $T \in \mathcal{T}$ in Algorithm 1 and Algorithm 3, and the $(k+2t)$-subset $\mathcal{R}$ of the n shares used in Algorithm 3, are also chosen randomly.

It turns out that the efficiencies of the three algorithms are very close. Although Algorithm 1 is usually a little more efficient than Algorithm 3, the expected number of shares needed in Algorithm 1 is much larger than that needed in Algorithm 3, especially for large n. Also note that s_{d_3} is smaller than $k + 2t$, the number of shares needed in McEliece and Sarwate [4].

Table 1.

No	n	k	t	β_{d_1}	β_r	β_{d_3}	s_{d_1}	s_{d_3}	$k+2t$
1	9	3	2	1.833	2.400	2.145	8.832	6.443	7
2	10	3	2	1.733	2.143	2.041	9.813	6.428	7
3	11	3	2	1.655	1.964	1.926	10.836	6.227	7
4	12	3	2	1.591	1.833	1.800	11.863	6.154	7
5	13	3	2	1.538	1.733	1.701	12.871	6.065	7
6	59	3	2	1.105	1.111	1.134	58.957	5.250	7
7	109	3	2	1.056	1.058	1.068	108.981	5.132	7
8	159	3	2	1.038	1.039	1.045	158.989	5.097	7
9	209	3	2	1.029	1.029	1.031	208.988	5.053	7
10	12	4	2	1.818	2.357	2.277	11.885	7.336	8
11	13	4	2	1.744	2.167	2.258	12.895	7.294	8
12	14	4	2	1.681	2.022	2.132	13.892	7.216	8
13	15	4	2	1.629	1.909	1.948	14.906	7.121	8
14	16	4	2	1.583	1.818	1.921	15.902	7.085	8
15	62	4	2	1.134	1.144	1.192	61.963	6.287	8
16	112	4	2	1.073	1.076	1.104	111.984	6.160	8
17	162	4	2	1.050	1.051	1.082	161.983	6.126	8
18	262	4	2	1.031	1.031	1.041	261.991	6.064	8
19	16	4	3	2.036	2.545	2.582	15.880	9.132	10
20	17	4	3	1.956	2.378	2.467	16.884	8.985	10
21	18	4	3	1.887	2.242	2.393	17.864	8.944	10
22	19	4	3	1.828	2.130	2.291	18.867	8.886	10
23	20	4	3	1.775	2.036	2.171	19.899	8.813	10
24	66	4	3	1.196	1.210	1.274	65.956	7.592	10
25	116	4	3	1.108	1.112	1.135	115.990	7.329	10
26	216	4	3	1.057	1.058	1.079	215.986	7.179	10
27	316	4	3	1.039	1.039	1.056	315.984	7.139	10

5 Conclusion

In this paper, we proposed an algorithm for the problem of determining a consistent set of shares in a threshold scheme with cheaters to reconstruct the secret correctly. This algorithm is a modification of Rees et al. 's [6] two algorithms but can reduce the number of shares to be pooled to $k+t$ in the best case, and less than or equal to $k+2t$ in other cases, while Rees et al. 's algorithms need all the good shares to be pooled so that the least number of shares is $n-t$ in the best case and n in other cases. The efficiencies of these three algorithms are similar but the shares needed in Rees et al. 's two algorithms are much more than those in ours, especially when n is large. In addition, the number of shares used in the newly proposed algorithm is usually less than $k+2t$, the number of shares needed in McEliece and Sarwate [4], which happens more often for large n. Therefore, there is a real advantage in using the newly proposed algorithm, especially in the case when gathering all the participants together is impossible, or in a (k, n) threshold scheme with small k and t but a relatively large n. Another advantage is that only one covering, the t-$(k+2t, 2t, 1)$ covering, is needed in any (k, n_s) multiple threshold scheme of s multiplicity, or in s different (k, n_s) threshold schemes with the same k and t but different n_s. As a consequence, this modified algorithm is shown to be practical.

Acknowledgements

This research is supported by Grant-in-Aid for Scientific Research (C) under Contract Number 14540100. The authors thank Professor Ruizhong Wei of Lakehead University for his helpful comments on an earlier version of this paper.

References

[1] G. R. Blakley, *Safeguarding cryptographic keys*, AFIPS Conf. Proceed. **48** (1979), 313–317. 377

[2] E. F. Brickell and D. R. Stinson, *The detection of cheaters in threshold schemes*, SIAM J. Discrete Math. **4** (1991), 502–510. 377

[3] L. Harn, *Efficient sharing (broadcasting) of multiple secrets*, IEE Proc. -Comput. Digit. Tech. **142** (1995), 237–240. 377

[4] R. J. McEliece and D. V. Sarwate, *On sharing secrets and Reed-Solomon codes*, Comm. ACM **24** (1981), 583–584. 377, 378, 381, 383, 384

[5] W. Ogata and K. Kurosawa, *Optimum secret sharing scheme secure against cheating*, Lecture Notes in Comput. Sci. **1070** (1996), 200–211. 377

[6] R. S. Rees, D. R. Stinson, R. Wei and G. H. J. van Rees, *An application of covering designs: determining the maximum consistent set of shares in a threshold scheme*, Ars Combin. **53** (1999), 225–237. 378, 379, 380, 381, 382, 384

[7] A. Shamir, *How to share a secret*, Comm. ACM **22** (1979), 612–613. 377

[8] M. Tompa and H. Woll, *How to share a secret with cheaters*, J. Cryptology **1** (1988), 133–138. 377

[9] URL: http://www.ccrwest.org/cover.html 383

Non-interactive Deniable Ring Authentication

Willy Susilo and Yi Mu

Centre for Information Security Research
School of Information Technology and Computer Science
University of Wollongong, Wollongong 2522, Australia
{wsusilo,ymu}@uow.edu.au

Abstract. In this paper, we propose a new primitive called *non interactive* deniable ring authentication: it is possible to convince a verifier that a member of an *ad hoc* collection of participants is authenticating a message m without revealing which one and the verifier V cannot convince any third party that the message m was indeed authenticated in a non-interactive way. Unlike the deniable ring authentication proposed in [19], we require this primitive to be *non-interactive*. Having this restriction, the primitive can be used in practice without having to use the anonymous routing channel (eg. MIX-nets) introduced in [19]. In this paper, we provide the formal definition of non-interactive deniable ring authentication schemes together with a generic construction of such schemes from any ring signature schemes. The generic construction can be used to convert any existing ring signature schemes, for example [20, 1], to non-interactive deniable ring authentication schemes. We also present an extension of this idea to allow a non-interactive *deniable ring to threshold ring* authentication. In this scenario, the signature can convince a group of verifiers, but the verifiers cannot convince any other third party about this fact, because any collusion of t verifiers can always generate a valid message-signature pair. We also present a generic construction of this scheme. A special case of this scenario is a *deniable ring-to-ring* authentication scheme, where the collaboration of *all* verifiers is required to generate a valid message-signature pair.

Keywords. Ring signature schemes, deniable, non-interactive, ring-to-ring authentication

1 Introduction

Consider a situation when Alice, who is a member of the parliament, wishes to inform the prime minister about very sensitive information related to the country. In this situation, Alice does not want her identity to be revealed by the prime minister, and on the other hand, she also wants the prime minister to keep this information for himself and not to be forwarded to any other person. To make the information reliable, it must be authenticated and this must be verifiable by the prime minister that it comes from one of the parliament's member, so that the prime minister can make his decision on this matter.

Alice cannot use a standard digitally signed message, since this message will directly reveal her identity to the prime minister. With the recent work introduced by Rivest, Shamir and Tauman [20], called *ring signature*, Alice's identity can be hidden and the message can be identified to come from one of the parliament members without revealing who the actual signer is, but the prime minister can quote this message and publish it as an authenticated message that comes from one of the parliament member - something that Alice does not want to happen.

Motivated by the above idea, the recent work introduced by Naor [19], called *Deniable Ring Authentication*, can be used. However, this scheme requires an interactive zero knowledge protocol, which makes it impractical.

The intention of this paper is to introduce a new notion called a *non-interactive deniable ring authentication*, which allows a signer to sign a message m on behalf of an ad hoc collection of participants, and to convince a designated verifier V that this message is correct. We note that in practice, we replace m with $H(m)$, where $H(\cdot)$ denote a collision-free hash function. Moreover, it is required that the designated verifier V cannot convince any other third party that the message m was indeed authenticated. We also require that a *non-interactive* verification must be used, because the ad hoc collection of participants might not be aware to acknowledge any authentication request whenever they are asked (c.f. [19]). In this situation, V can verify the correctness of the message whenever he would like to do so, but he cannot show this message to anyone else and claim that this was authenticated by the ad hoc collection of participants.

Roughly speaking, for a scheme to be Non-interactive Deniable Ring Authentication, it should: (i) Enable a sender in an ad hoc collection $\mathbb{S}$ and any message m to prove (non-interactively) that a member of $\mathbb{S}$ is the one authenticating the signature. (ii) Be a good authentication scheme, i.e. not allowing forgeries, where the notions of forgeability of Goldwasser, Micali and Rivest [14] are relevant. (iii) The authentication is *deniable* in the zero-knowledge sense. The recipient could have generated the conversation alone and the result would have been indistinguishable. (iv) The authentication must be *signer-ambiguous.* (v) The verifier is not part of the system.

We provide a formal model of a non-interactive deniable ring authentication scheme. We also propose a generic method for converting a ring signature scheme into a non-interactive deniable ring authentication scheme. We show that a non-interactive deniable ring authentication scheme can be constructed from a ring signature scheme combined with a chameleon hash [18]. We also extend our work to provide a non-interactive deniable ring to threshold ring authentication scheme. In this scenario, the signer in the ring can sign a message and convince a group of verifiers about this fact, but the verifiers cannot convince any third party about the authenticity of this message, because a collusion of t verifiers can always create a valid message-signature pair that will also pass the verification stage. We also show a generic construction of this type of scheme, together with a special case when $t = n$, which we call *deniable ring-to-ring* authentication scheme.

1.1 Related Work

In [20], the definition of *ring signatures* was formalized and an efficient scheme based on RSA was proposed. A ring signature scheme is based on trapdoor one-way permutations and an ideal block cipher that is regarded as a perfectly random permutation. A ring signature scheme allows a signer who knows at least one secret information (or trapdoor information) to produce a sequence of n random permutation and form them into a ring. This signature can be used to convince any third party that one of the people in the group (who knows the trapdoor information) has authenticated the message on behalf of the group. The authentication provides *signer ambiguity*, in the sense that no one can identify who has actually signed the message.

In [1], a method to construct a ring signature from different types of public keys, such as these for integer factoring based schemes and discrete log based schemes, was proposed. The proposed scheme is more efficient than [20]. The formal security definition of a ring signature is also given in [1].

In [19], the notion of ring signatures was combined with deniable authenticaton [13]. The result is called *Deniable Ring Authentication* that allows a signer to authenticate a message m on behalf of an ad hoc collection of users and to convince a verifier that this authentication is done correctly. Moreover, the verifier cannot convince any third party that the message m was indeed authenticated. There is no 'paper trail' of the conversation, other than what could be produced by the verifier alone, as in zero-knowledge [19]. However, the verification is done interactively, and hence, the requirement of having an anonymous routing, such as MIX-nets, is essential. Moreover, as a result of the requirement of this new notion, the message size is longer compared to a normal ring signature.

Another related notion is group signature schemes. These schemes allow members of a *fixed* group to sign messages on the group's behalf while their anonymity is preserved. This anonymity is conditional and the group manager can always revoke it. This concept was introduced by Chaum in [11]. The original scheme has the characteristic that the size of the signature is always proportional to the group size. However, since the introduction of the scheme in [6], the size of the signature is fixed and it does not depend on the group size.

Designated Verifier Proofs were proposed in [17]. The idea is to allow signatures to convince only the intended recipient, who is assumed to have a public-key. As noted in [20], ring signature schemes can be used to provide this mechanism by joining the verifier in the ring. However, it might not be practical in the real life since the verifier might not have any public key setup.

Dwork, Naor and Sahai proposed *deniable authentication* in [13]. Deniable authentication provides a system that addresses the deniability aspects, i.e. the protocol does not leave any paper trail for the authentication of the message. This work allows a single signer to achieve this property.

Undeniable signatures [10] allows a signature to be verified by everyone but requires the help of the signer. The signer is able to reject invalid signatures, but he must not be able to deny valid signatures. If the signer is unavailable or unwilling to cooperate, the signature would not be longer verifiable. To overcome this shortcoming, the notion of *confirmer signature* [9] is proposed. In confirmer signatures, the ability to verify or deny signatures is transferred to a designated confirmer. A generic construction of a confirmer signature from an ordinary signature scheme is proposed in [7].

The notion of *Universal Designated-Verifier* Signatures (UDVS) was recently proposed in [23]. A UDVS scheme can be used as a standard publicly-verifiable digital signature but has additional functionality that allows any signature holder to designate the signature to any desired designated verifier, using the verifier's public key. The construction proposed in [23] is based on bilinear pairing, which is an extension of Boneh-Lynn-Shacham (BLS) short signature scheme [4].

Our Contributions

We define the notion of *non-interactive deniable ring authentication.* Our schemes provide a ring signature scheme with a designated verifier. The verification is done non-interactively by the verifier. We note that the idea of converting interactive protocols to non-interactive ones is not new, for example as suggested in [12]. Our schemes will not require any additional requirement such as an anonymous routing (cf. [19]). We provide a generic construction of non-interactive deniable ring authentication. The size of the resulting new scheme is the same as the original ring signature scheme plus a random number (cf. [19] which requires about twice the size of the message). We only require the designated verifier V to have a published chameleon hash function to be used by the signer. We note that this requirement might be removed in a different construction. We also extend our basic schemes to *deniable ring to ring* authentication schemes. In these schemes, a message authenticated by a member of a ring $\mathbb{S}$ can be designated to a group of people, $\mathbb{V}$. The notion of *designation* in this context refers to the inability of the verifiers in $\mathbb{V}$ to convince a third party outside $\mathbb{V}$ about this fact because the verifiers themselves could produce such signature, by collaborating with the other verifiers in $\mathbb{V}$. When t verifiers in $\mathbb{V}$ are required to collaborate to generate a new valid message-signature pair, we call this scheme a *deniable ring to threshold ring* authentication scheme. When *all* verifiers in $\mathbb{V}$ are required to collaborate to generate a valid message-signature pair, we call it a *deniable ring-to-ring* authentication scheme. We show how to construct such schemes from *any* existing ring signature schemes.

1.2 Cryptographic Tools

Chameleon Hashing

Chameleon hashing is basically non-interactive commitment schemes as proposed by Brassard, Chaum and Crepeau [5]. The idea of chameleon hash functions was introduced and formalized in [18] in the construction of their chameleon signature schemes. The name "chameleon" refers to the ability of the owner of the trapdoor information to change the input to the function to any value of his choice without changing the resulting output.

A chameleon hash function is associated with a pair of public and private keys and has the following properties [18]: (1) Anyone who knows the public key can compute the associated hash function. (2) For people who do not have the knowledge of the trapdoor (i.e. the secret key), the hash function is collision resistant: it is infeasible to find two inputs which are mapped to the same output. (3) The trapdoor information's holder can easily find collisions for every given input.

Several constructions of chameleon hashing have been proposed in [18], which are based on discrete log and [8], which is based on the hardness of deciding whether an element is a "small" e-th residue modulo N^2.

Access Structures and Cumulative Arrays

Let $\mathbb{V} = \{\mathsf{V}_1, \cdots, \mathsf{V}_n\}$ be a group of n players and let $2^{\mathbb{V}}$ denote the family of all subsets of $\mathbb{V}$. A subset Γ of $2^{\mathbb{V}}$ with the property that if $\mathcal{A} \in \Gamma$ and $\mathcal{A} \subseteq \mathcal{A}'$ then $\mathcal{A}' \in \Gamma$ is called *monotone increasing*. An access structure Γ is a monotone increasing subset of $2^{\mathbb{V}}$. Elements in Γ are called *authorized subsets* on Γ and elements not in Γ are called *unauthorized subset*. The notion of access structures plays an important role in the theory of secret sharing.

The first (t, n) threshold secret sharing scheme independently invented by Blakley [2] and Shamir [21] in which the access structure is defined by

$$\Gamma = \{\mathcal{A} \subseteq \mathbb{V} || \mathcal{A}| \geq t\}$$

and is known as the (t, n) *threshold access structure*.

Cumulative maps (or also called *cumulative arrays*) for access structures were formally introduced in [22], but the idea behind them was implicitly used in the constructions given in [15]. The construction of *minimal* cumulative maps for any access structures were developed in [16].

The rest of this paper is organized as follows. In the next section, we give a model for a non-interactive deniable ring authentication scheme and outline its security requirements. Section 3 proposes a generic construction of non-interactive deniable ring authentication schemes from ring signature schemes and chameleon hashing. In section 4, we present an extension of the basic scheme to allow a deniable ring to threshold ring authentication schemes. We also provide a generic construction of such schemes. Finally, we provide a special case of deniable ring to threshold ring authentication schemes called *deniable ring-to-ring* authentication schemes. Section 5 concludes the paper.

2 Non-interactive Deniable Ring Authentication

In this section, we provide a definition of non-interactive deniable ring authentication. This authentication can also be referred to as ring authentication with designated verifier. We now provide the summary of the setup and requirements of a non-interactive deniable ring authentication.

Setup. We assume that the participants have published their public keys. The public keys are generated via a standard public key generation algorithm. We define the *ring* as follows.

A ring $\mathbb{S}$ contains any subset of participants. An authenticator $\mathsf{S}_i \in \mathbb{S}$ can sign on behalf of $\mathbb{S}$. The verifier of a message, V, is an arbitrary party. We require that $\mathsf{V} \notin \mathbb{S}$. We also require that V has published a hash function $\mathcal{H}$ that can be used by the signers $\mathbb{S}$. We assume that both verifier and the authenticator have access to the public keys of all members $\mathsf{S}_i \in \mathbb{S}$. The verifier V can verify an authenticated message non-interactively, without the help of any member in $\mathbb{S}$. We require the authentication scheme to provide:

– **Completeness:** For any subset of participants $\in \mathbb{S}$ and for any good authenticator $\mathsf{S}_i \in \mathbb{S}$, for any message $m \in \{0,1\}^*$, if S_i has followed the signature generation protocol correctly, and V has also followed the verification algorithm correctly, then V accepts with an overwhelming probability.
– **Signer Ambiguity:** For any participant $\mathsf{S}_i \in \mathbb{S}$ and assuming the size of $|\mathbb{S}| = \hat{s}$, the probability that an authenticated message m is signed by S_i is at most $\frac{1}{\hat{s}}$.
– **Soundness:** The soundness of non-interactive deniable ring authentication scheme is computational, since the underlying ring signature scheme cannot be stronger than the individual signature scheme used by the possible signers. We consider an attack to be successful, if after analyzing n non-interactive deniable ring authentication, the forger can construct a valid non-interactive deniable ring authentication on behalf of the same group, for a different message $m \notin \{m_i\}_{i=1,2,\cdots,n}$, where the verifier V will accept.

In the following definition, we denote $< s_{k_i}, p_{k_i} >$ as a pair of secret and public key according to a specific algorithm, that is owned by S_i. A non-interactive deniable authentication scheme consists of the following algorithms:

– $\texttt{DSign}(m, s_k, L)$**:** is a probabilistic polynomial time algorithm that takes a message $m \in \{0,1\}^*$ and a list L that contains a set of public keys, including the one that corresponds to the secret key, s_k, and outputs a signature σ.
– $\texttt{DVerify}(m, \sigma, L)$**:** is a deterministic non-interactive polynomial-time algorithm that takes a message m, a signature σ and a list of public keys L, and outputs either `True` or $\perp$ meaning `accept` or `reject`, respectively. We require that

$$\texttt{Pr}(\{m, \sigma, L\} : \sigma \leftarrow \texttt{DSign}(m, s_k, L); \texttt{True} \leftarrow \texttt{DVerify}(m, \sigma, L)) = 1$$

L includes public keys based on different security parameters, and the security of $\texttt{DSign}(m, s_k, L)$ is set to the smallest one among them. L can include several types of public-keys at the same time, such as for RSA and Schnorr in a particular construction.

We require that the probability

$$\mathtt{Pr}\left[\begin{array}{l}\hat{\sigma} \leftarrow \mathtt{DSign}(m, s_{k_i}, L), \\ \qquad L := \{p_{k_1}, p_{k_2}, \cdots, p_{k_n}\}, \\ \qquad i \notin \{1, 2, \cdots, n\}, \\ \qquad \mathtt{True} \leftarrow \mathtt{DVerify}(m, \hat{\sigma}, L)\end{array}\right]$$

is negligible.

We also require that a non-interactive deniable ring authentication scheme must provide an *existential unforgeability*. Unforgeability requires that it be difficult to forge a valid non-interactive deniable ring authenticated message. The advantage in existential forging a non-interactive deniable ring authenticated message of an algorithm $\mathcal{F}$, given access to a non-interactive deniable ring authentication signature generation oracle S, along with a hash oracle, is

$$\mathtt{AdvDSigF}_{\mathcal{F}} \overset{\mathtt{def}}{=} \mathtt{Pr}\left[\begin{array}{l}\mathtt{DVerify}(m, \sigma, < p_{k_1}, p_{k_2}, \cdots, p_{k_n} >) = \mathtt{True}: \\ \qquad \{< p_{k_1}, s_{k_1} >, < p_{k_2}, s_{k_2} >, \\ \qquad\quad \cdots, < p_{k_n}, s_{k_n} >\} \overset{R}{\leftarrow} \mathtt{KeyGen}(), \\ \qquad \sigma \overset{R}{\leftarrow} \mathcal{F}^S(m, s_{k_i}, < p_{k_1}, p_{k_2}, \cdots, p_{k_n} >)\end{array}\right]$$

The probability is taken over the coin tosses of the key generation algorithms, of the oracles, and of the forger. The forger is additionally constrained in that its forgery on a message m must be non trivial. It must not previously have queried the oracle S at m.

Definition 1. *A non-interactive deniable ring authentication signature forger $\mathcal{F}(t, q_H, q_S, \epsilon)$-forges a non-interactive deniable ring authentication signature if: Algorithm $\mathcal{F}$ runs in time at most t; $\mathcal{F}$ makes at most q_H queries to the hash function, at most q_S queries to the oracle S; and $\mathtt{AdvDSigF}_{\mathcal{F}}$ is at least ϵ. A non-interactive deniable ring authentication scheme is (t, q_H, q_S, ϵ)-secure against existential forgery if no forger (t, q_H, q_S, ϵ) breaks it.*

3 Generic Construction of Non-Interactive Deniable Ring Authentication Schemes

In this section, we provide a generic construction for constructing non-interactive deniable ring authentication schemes from ring signature schemes. Before proceeding with the generic construction, we will review the ring signature schemes and the chameleon hashing to highlight the notations that will be used in this section.

3.1 Ring Signature Schemes

We use the notation proposed in [1] to define ring signature schemes. We note that the ring signature schemes are referred to 1-out-of-n in [1].

Definition 2. *[1] A ring signature scheme consists of three polynomial time algorithms*

– $(s_k, p_k) \leftarrow \mathcal{G}(1^\kappa)$*: A probabilistic algorithm that takes security parameter* κ *and outputs private key* s_k *and public key* p_k*.*
– $\sigma \leftarrow \mathcal{S}(m, s_k, L)$*: A probabilistic algorithm that takes a message* m*, a list* L *that contains public keys including the one that corresponds to* s_k *and outputs a signature* σ*.*
– $\{\texttt{True} \text{ or } \perp\} \leftarrow \mathcal{V}(m, \sigma, L)$*: A deterministic algorithm that takes a message* m *and a signature* σ*, and outputs either* `True` *or* $\perp$ *meaning* accept *or* reject*, respectively. It is required to have* $\texttt{True} \leftarrow \mathcal{V}(m, \mathcal{S}(m, s_k, L), L)$ *with an overwhelming probability.*

A ring that allows a mixture of factorization and discrete log based public keys has been constructed in [1].

3.2 Chameleon Hashing

A chameleon hash function is associated with a user U_i who has published a public hashing key, $\mathcal{H}_{U_i}$, and holds the corresponding secret key (i.e. the trapdoor for finding collisions), denoted by $s_{k_{U_i}}$. We abuse the notation s_k in this context, because s_k is basically a secret key that is generated via a key generation algorithm by U_i, and the public (hashing) key $\mathcal{H}_{U_i}$ defines the chameleon hash function, denoted by $\texttt{CHAM\_HASH}_{U_i}(\cdot,\cdot)$, which can be computed efficiently given the value of $\mathcal{H}_{U_i}$ [18].

The properties of $\texttt{CHAM\_HASH}_{U_i}(\cdot,\cdot)$ are illustrated as follows [18].

– On input a message m and a random string r, $\texttt{CHAM\_HASH}_{U_i}(m, r)$ generates a hash value h.
– The hash value $h \leftarrow \texttt{CHAM\_HASH}_{U_i}(m, r)$ satisfies the following properties.
  - **Collision Resistance:** There is no efficient algorithm that on input the public key $\mathcal{H}_{U_i}$ can find pairs (m_1, r_1) and (m_2, r_2) where $m_1 \neq m_2$ such that $\texttt{CHAM\_HASH}_{U_i}(m_1, r_1) = \texttt{CHAM\_HASH}_{U_i}(m_2, r_2)$ with a non-negligible probability.
  - **Trapdoor Collisions:** Given the secret trapdoor information s_k, together with (m_1, r_1), there is an efficient algorithm that can find an arbitrary message m_2 such that $\texttt{CHAM\_HASH}_{U_i}(m_1, r_1) = \texttt{CHAM\_HASH}_{U_i}(m_2, r_2)$ with an overwhelming probability.
  - **Uniformity:** For a randomly chosen r, all messages m induce the same probability distribution on $\texttt{CHAM\_HASH}_{U_i}(m, r)$.

Concrete examples of chameleon hash functions have been developed in [18] and [8]. In the following, we recall the chameleon hash function constructed in [18], which is based on the hardness of discrete log.

In the setup phase, U_i generates two prime numbers p and q such that $q|p-1$, where q is a large enough prime factor. Then, an element g where $ord_p(g) = q$ is chosen, from $\mathbb{Z}_p^*$. The private key s_k is a random $x \in \mathbb{Z}_q^*$. The public key $\mathcal{H}_{U_i}$ is

$y = g^x \pmod p$. We note that p, q, g are implicit parts of the public key. The chameleon hash function $\texttt{CHAM\_HASH}_{U_i}(m, r)$ is defined as follows. For a given message $m \in \mathbb{Z}_q^*$, choose a random value $r \in \mathbb{Z}_q^*$, and set $\texttt{CHAM\_HASH}_{U_i}(m, r) = g^m y^r \pmod p$.

We note that the above chameleon hash function is collision resistant for any $U_j \neq U_i$. U_i can always find any other message $\hat{m} \neq m$, and compute the appropriate $\hat{r}$ to find the same hash value, because he knows x and he can easily solve $m + xr = \hat{m} + x\hat{r} \pmod q$.

3.3 Generic Construction for Non-Interactive Deniable Ring Authentication

In this section, we proceed with our generic construction for non-interactive deniable ring authentication from any ring signature schemes. The conversion is defined as follows. (Let $\texttt{CHAM\_HASH}_{\mathsf{V}}(\cdot, \cdot)$ be the chameleon hash function published by the verifier V).

1. Define:
$$\texttt{DSign}(m, s_k, L) \stackrel{\Delta}{=} \begin{cases} h \leftarrow \texttt{CHAM\_HASH}_{\mathsf{V}}(m, r), \text{for a random } r \\ \sigma_1 \leftarrow \mathcal{S}(h, s_k, L), \\ \sigma \leftarrow (\sigma_1 || r) \end{cases}$$
We define the signature of message m to be σ.
2. Define:
$$\texttt{DVerify}(m, \sigma, L) \stackrel{\Delta}{=} \begin{cases} (\sigma_1 || r) \leftarrow \sigma, \\ h \leftarrow \texttt{CHAM\_HASH}_{\mathsf{V}}(m, r), \\ \texttt{Result} \leftarrow \mathcal{V}(h, \sigma_1, L) \end{cases}$$
The result of the verification is defined as $\texttt{Result} \leftarrow \texttt{DVerify}(m, \sigma, L)$, which is either $\texttt{True}$ or $\perp$, meaning $\texttt{accept}$ or $\texttt{reject}$, respectively.

The resulting signature is non-transferable. We note that the resulting non-interactive deniable ring authentication does not allow the verifier V to convince any third party about this fact, due to the use of chameleon hashing in the above scheme. Since V knows the secret key s_k used in $\texttt{CHAM\_HASH}_{\mathsf{V}}(m, r)$, then he can always generate another pair $(\hat{m}, \hat{r})$, for $\hat{m} \neq m$ that will pass the verification $\texttt{DVerify}(\hat{m}, \sigma, L)$ for the same signature σ.

The resulting signature provides signer-ambiguity. It is straight forward to see that the signer-ambiguity is provided by the original ring signature used in the conversion.

The resulting non-interactive deniable ring authentication scheme provides an existential unforgeability iff the underlying ring signature scheme used in the conversion provides an existential unforgeability. It is straight forward to see that the existential unforgeability of the underlying ring signature scheme will result in the existential unforgeability of the resulting non-interactive deniable ring authentication scheme.

3.4 An Example

We present a sample conversion of the ring signature scheme proposed in [3] to make a non-interactive deniable ring authentication scheme as described in previous section.

We follow the settings of the original ring signature proposed in [3]. There is a set U of users, where each user $u \in U$ has a signing key pair (PK_u, SK_u). The ring signature is constructed from bilinear maps. Recall g_1, g_2 are generators of group G_1, G_2 respectively, and $e : G_1 \times G_2 \to G_T$ is a bilinear map, and a computable isomorphism $\psi : G_1 \to G_2$ exists, with $\psi(g_1) = g_2$. There is a full-domain hash function $h : \{0,1\}^* \to G_2$. The security analysis considers h as a random oracle [3].

In the following, we use the chameleon hash function $\texttt{CHAM\_HASH}_{\mathsf{V}}(m, r) = g^m y^r \pmod p$ as defined in [18]. We assume the verifier V publishes this chameleon hash function, and keeps the secret key $x = \log_g(y)$. The non-interactive deniable ring authentication is defined as follows.

Key Generation. For any user $u \in U$, pick a random $\bar{x} \in_R \mathbb{Z}_p$ and compute $v = g_1^{\bar{x}}$. The user's public key is $v \in G_1$, and the user's secret key is $\bar{x} \in \mathbb{Z}_p$.

Non-Interactive Ring Signing. Given public keys $v_1, \cdots, v_n \in G_1$, a message $m \in \mathbb{Z}_p$ and a private key $\bar{x}$ corresponding to one of the public keys v_s for some s, do the following.

1. Choose a random $r \in \mathbb{Z}_p$ and compute $\hat{h} = h(\texttt{CHAM\_HASH}_{\mathsf{V}}(m, r))$
2. Choose random $a_i \in \mathbb{Z}_p$ for all $i \neq s$.
3. Set
$$\sigma_s = \left(\hat{h} / \psi \left(\prod_{i \neq s} v_i^{a_i} \right) \right)^{1/\bar{x}}$$
4. For all $i \neq s$, set $\sigma_i = g_2^{a_i}$.

The signature is (m, r, σ), where $\sigma =< \sigma_1, \sigma_2, \cdots, \sigma_n >\in G_2^n$.

Non-Interactive Ring Verification. Given public keys $v_1, \cdots, v_n \in G_1$, a message $m \in \mathbb{Z}_p$, a ring signature σ, and r, compute $\hat{h} = h(\texttt{CHAM\_HASH}_{\mathsf{V}}(m, r))$ and verify that $e(g, h) = \prod_{i=1}^{n} e(v_i, \sigma_i)$.

Completeness. It is easy to show a signature produced by the non-interactive ring signing algorithm will verify under the non-interactive ring verification algorithm using the bilinearity and nondegeneracy of the pairing e.

Designated Signature. It is also easy to see that the verifier V can be convinced with the correctness of the signature σ, but no one else can be convinced with this fact because V can modify the message m with any message $m' \neq m$ of his choice, and find the appropriate $r' \neq r$, since V knows the secret key $x = \log_g(y)$.

4 An Extension: Non-interactive Deniable Ring to Threshold Ring Authentication Schemes

Consider a case where a member of an *ad hoc* group would like to reveal an information on behalf of the group to a designated group of people. We call this case as a *deniable ring to ring* authentication. We require that the people in the designated group cannot forward the message to convince a third party about this fact. When t verifiers (or recipients) are required to collaborate to forge the message-signature pair, we call this *deniable ring to threshold* (t, n) *ring authentication*. We define the *deniable ring to threshold* (t, n) *ring authentication* schemes as follows.

Definition 3. Deniable ring to threshold (t, n) ring authentication schemes (DTTR (t, n)) *allow any participants in a ring* $\mathbb{S}$ *to authenticate a message* $m \in \{0,1\}^*$ *on behalf of an ad hoc group* $\mathbb{S}$ *to convince a group of verifiers* $\mathbb{V}$ *of the authenticity of* m*, where* $\mathbb{S} \cap \mathbb{V} = \emptyset$*. However, the verifiers cannot convince any third party out of* $\mathbb{V}$ *about this fact, because any* t *out of* n *verifiers in* $\mathbb{V}$ *can create the message-signature pair by themselves.*

Definition 4. *In a* DTTR (t, n) *scheme, if* $n - t + 1$ *verifiers agree that they have not altered the authenticated message, then they can be convinced that the message is indeed authenticated by the ring* $\mathbb{S}$.

The *setup* procedure of DTTR (t, n) schemes is illustrated as follows. We assume that the participants have published their public keys. The public keys are generated via a standard public key generation algorithm.

A ring $\mathbb{S}$ contains any subset of participants, $\mathbb{S} = \{\mathsf{S}_1, \cdots, \mathsf{S}_\ell\}$. The verifier of the message, $\mathbb{V}$, is a set of participants, $\{\mathsf{V}_1, \mathsf{V}_2, \cdots, \mathsf{V}_n\} \in \mathbb{V}$, where $\mathbb{S} \cap \mathbb{V} = \emptyset$. We require that each verifier V_i has published a chameleon hash function $\mathcal{H}_i$ (that can be used by the signer $\in \mathbb{S}$). We assume that the verifiers and the authenticators have access to the public keys of all members $\mathsf{S}_i \in \mathbb{S}$ and $\mathsf{V}_i \in \mathbb{V}$. Each signer $\mathsf{S}_i \in \mathbb{S}$ can sign a message on behalf of the ring $\mathbb{S}$, and this message will be designated to $\mathbb{V}$. Each verifier $\mathsf{V}_i \in \mathbb{V}$ can verify an authenticated message non-interactively, without any help from $\mathbb{S}$. We require the authenticated message to be *designated*, which means that no third party outside $\mathbb{V}$ can be convinced with the fact that the authenticated message has indeed been signed by $\mathbb{S}$. The reason behind this is due to the fact that t-out-of-n verifiers can collaborate and create any valid message-signature pair themselves.

As in the standard definition of ring signature schemes, we require no setup procedure for the signers $\mathbb{S}$ to form the ring, as well as no setup procedure for the construction of the verifiers $\mathbb{V}$. We note that this requirement enables these schemes to be used in an ad hoc environment.

4.1 Constructing Non-interactive DTTR (t, n) Schemes

Inspiring by the original idea mentioned earlier, we can obtain a non-interactive DTTR (t, n) scheme as follows. We incorporate the idea of cumulative array [16, 22] in this scheme. The signature generation phase consists of two stages: 1) hash function generation and 2) signature generation.

Hash Function Generation
Step 1. Generating the Cumulative Array
We assume that each verifier $\mathsf{V}_i \in \mathbb{V}$ has published a chameleon hash function $\texttt{CHAM\_HASH}_{\mathsf{V}_i} = g^m y_i^r \pmod p$ as before. Each verifier V_i holds the secret key associated with $x_i = \log_g(y_i)$.

To construct a non-interactive $\texttt{DTTR}$ (t, n) scheme, firstly the signer in $\mathbb{S}$ generates a cumulative array for Γ, where Γ denotes *access structure*, i.e. the set of authorized participants. A cumulative scheme for Γ is a map $\alpha : \mathbb{V} \to 2^{\mathcal{F}}$, where $\mathcal{F}$ is a finite set such that for any $\mathcal{A} \subseteq \mathbb{V}$,

$$\bigcup_{\mathsf{V}_i \in \mathcal{A}} \alpha_i = \mathcal{F} \Longleftrightarrow \mathcal{A} \in \Gamma$$

where $\alpha_i = \alpha(\mathsf{V}_i)$. We can represent the scheme as a $|\mathbb{V}| \times |\mathcal{F}|$ cumulative array $\mathcal{C}(\Gamma) = [c_{ij}]$, where each entry c_{ij} is either 0 or 1 such that

$$c_{ij} = 1 \Longleftrightarrow \mathsf{V}_i \subseteq \mathcal{A}_j$$

Example 1. Suppose $\mathbb{V} = \{\mathsf{V}_1, \mathsf{V}_2, \mathsf{V}_3, \mathsf{V}_4\}$ and $\mathsf{S}_i \in \mathbb{S}$ would like to set up $(2, 4)$ scheme. Then, he will define the following cumulative array,

	$\mathsf{V}_1\mathsf{V}_2$	$\mathsf{V}_1\mathsf{V}_3$	$\mathsf{V}_1\mathsf{V}_4$	$\mathsf{V}_2\mathsf{V}_3$	$\mathsf{V}_2\mathsf{V}_4$	$\mathsf{V}_3\mathsf{V}_4$
V_1	1	1	1	0	0	0
V_2	1	0	0	1	1	0
V_3	0	1	0	1	0	1
V_4	0	0	1	0	1	1

We note that for each column $\mathsf{V}_i\mathsf{V}_j$, $i \neq j$, there are exactly two 1's, because the threshold $t = 2$ in the above example.

Example 2. For the above example, if $\mathsf{S}_i \in \mathbb{S}$ would like to setup $(3, 4)$ scheme, the cumulative array is defined as follows.

	$\mathsf{V}_1\mathsf{V}_2\mathsf{V}_3$	$\mathsf{V}_1\mathsf{V}_2\mathsf{V}_4$	$\mathsf{V}_1\mathsf{V}_3\mathsf{V}_4$	$\mathsf{V}_2\mathsf{V}_3\mathsf{V}_4$
V_1	1	1	1	0
V_2	1	1	0	1
V_3	1	0	1	1
V_4	0	1	1	1

Step 2. Generating the hash function
Next, the columns of the cumulative array are renamed to r_i, $i = 1, \cdots, \binom{n}{t}$. Finally, the hash function is defined as

$$\mathcal{H}(m, r_1, \cdots, r_{\binom{n}{t}}) = g^m \prod_{i=1}^{n} y_i^{\sum_{(\forall j, c_{ij}=1)} r_j} \pmod p$$

Example 3. In Example 1, the cumulative table is rewritten as

	V_1V_2	V_1V_3	V_1V_4	V_2V_3	V_2V_4	V_3V_4
	r_1	r_2	r_3	r_4	r_5	r_6
V_1	1	1	1	0	0	0
V_2	1	0	0	1	1	0
V_3	0	1	0	1	0	1
V_4	0	0	1	0	1	1

and the hash function $\mathcal{H}(m, r_1, r_2, r_3, r_4, r_5, r_6)$ is defined as

$$\mathcal{H}(m, r_1, r_2, r_3, r_4, r_5, r_6) = g^m y_1^{r_1+r_2+r_3} y_2^{r_1+r_4+r_5+} y_3^{r_2+r_4+r_6} y_4^{r_3+r_5+r_6} \pmod p$$

Similarly, the hash function in Example 2 is defined as

$$\mathcal{H}(m, r_1, r_2, r_3, r_4) = g^m y_1^{r_1+r_2+r_3} y_2^{r_1+r_2+r_4} y_3^{r_1+r_3+r_4} y_4^{r_2+r_3+r_4} \pmod p$$

Signature Generation
After defining the hash function $\mathcal{H}(m, r_1, \cdots, r_{\binom{n}{t}})$, $\mathsf{S}_i \in \mathbb{S}$ generates a signature as follows (note: L is the collection of public keys of $\mathbb{S}$)

$$\begin{aligned} h &\leftarrow \mathcal{H}(m, r_1, \cdots, r_{\binom{n}{t}}), \text{for randomly chosen } r_1, \cdots, r_{\binom{n}{t}} \\ \sigma_1 &\leftarrow \mathcal{S}(h, s_{k_i}, L) \\ \sigma &\leftarrow (\sigma_1 || r_1 || \cdots || r_{\binom{n}{t}})) \end{aligned}$$

The signature is sent together to the verifiers. We note that the definition of the hash function $\mathcal{H}(m, r_1, \cdots, r_{\binom{n}{t}})$ is public.

Signature Verification
To verify a signature, anyone can perform the following

$$\begin{aligned} (\sigma_1 || r_1 || \cdots || r_{\binom{n}{t}}) &\leftarrow \sigma \\ h &\leftarrow \mathcal{H}(m, r_1, \cdots, r_{\binom{n}{t}}) \\ \texttt{Result} &\leftarrow \mathcal{V}(h, \sigma_1, L) \end{aligned}$$

The result of the verification is either `True` or $\perp$, meaning `accept` or `reject`, respectively.

The soundness and completeness of the scheme can be derived easily as before.

Theorem 1. *In a non-interactive* `DTTR` (t, n) *scheme defined above, the verifiers* $\mathbb{V}$ *cannot convince any third party about the authenticity of the signature* σ *because any collusion of* t*-out-of-*n *verifiers can create any valid message-signature pair by themselves.*

Proof. From the definition of the hash function $\mathcal{H}(m, \cdot)$ above, a collusion of any t-out-of-n verifiers in $\mathbb{V}$ (who knows the associated secret keys $x_i = \log_g(y_i)$) can always find a valid message-signature pair, after seeing a valid message-signature pair, since they have a common r_i associated in the hash function. To be more precise, we will illustrate this as follows.

Consider a valid message-signature pair (m, σ) that was signed by $\mathsf{S}_i \in \mathbb{S}$ for verifiers $\{\mathsf{V}_1, \cdots, \mathsf{V}_n\}$. The message was signed using the following hash function

$$\mathcal{H}(m, r_1, \cdots, r_j, \cdots, r_{\binom{n}{t}}) = g^m \prod_{i=1}^{n} y_i^{\sum_{(\forall k, c_{ik}=1)} r_k} \pmod{p}$$

for $j \in \{1, \cdots, \binom{n}{t}\}$. We note that the verifiers obtain the following equation

$$\gamma = m + \sum_{i=1}^{n} \left[x_i \left(\sum_{(\forall k, c_{ik}=1)} r_k \right) \right] \pmod{q}$$

for an integer γ. However, t colluders only know t secret keys $x_i = \log_g(y_i)$. We also note that due to the construction of the cumulative array $\mathcal{C}(\Gamma)$, any t out of n verifiers will share a common r_j. Therefore, the colluders can also obtain

$$\hat{\gamma} = m + \left[\sum_{\forall i, \mathsf{V}_i \text{ colludes}} x_i r_j \right] \pmod{q}$$

for $\hat{\gamma} \in \mathbb{Z}_q$. Knowing the value of r_j and m, they can easily compute $\hat{\gamma}$, and choose any message $m' \neq m$ with the associated $r'_j \neq r_j$. It is easy to see that the new message-signature pair, (m', σ), where σ contains $r_1, r_2, \cdots, r'_j, \cdots, r_{\binom{n}{t}}$, is valid under the same verification algorithm. □

4.2 A Special Case: Non-interactive Deniable Ring-to-Ring Authentication Schemes

A special case of the non-interactive `DTTR` (t, n) scheme is the non-interactive deniable ring to (n, n) ring authentication, or simply called *deniable ring-to-ring* authentication. In this scheme, the setting is the same as the `DTTR` (t, n) scheme. However, if the verifiers would like to create another valid message-signature pair, then a collusion of n (or all) verifiers in $\mathbb{V}$ is required.

To design a ring-to-ring authentication, we can use the same method as before. However, since we require that only the collusion of *all* verifiers that can create any valid message-signature pair, then the hash function can be simplified as follows.

$$H(m, r) = g^m y_1^r y_2^r \cdots, y_n^r \pmod{p}$$

We note that according to Definition 4, any verifiers in $\mathbb{V}$ can be convinced with the authenticity of the message if he/she has not modified the message m.

5 Conclusion

We introduced *Non-Interactive Deniable Ring Authentication* schemes to allow a user in an ad hoc mode to designate his signature to a third party. We defined precise security notions for such schemes and proposed a way to convert *any* ring signature schemes into such schemes. We note that in our construction, the length of the signature is the same as the length of the underlying ring signature used in the conversion plus a random number (cf. [19]). We also presented an extension of this notion, namely *Non-Interactive Deniable Ring to Threshold Ring Authentication* that allows a user in ad hoc mode to designate his signature to a group of verifiers. However, the verifiers cannot convince any other third party about this fact, because any collusion of t verifiers can always produce a valid message-signature pair. We also show a special case when $t = n$ that we call *Deniable Ring-to-Ring Authentication* schemes.

Acknowledgement

We would like to thank the anonymous referees whose comments and suggestions helped us to improve this paper.

References

[1] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n Signatures from a Variety of Keys. *Advances in Cryptology - Asiacrypt 2002, Lecture Notes in Computer Science 2501*, pages 415 – 432, 2002. 386, 388, 392, 393

[2] G. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313 – 317, 1979. 390

[3] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiable Encrypted Signatures from Bilinear Maps. *Proceedings of Eurocrypt 2003, Lecture Notes in Computer Science 2656*, pages 416 – 432, 2003. 395

[4] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. pages 514–532. Springer Verlag, 2001. 389

[5] G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *JCSS, 37(2)*, pages 156 – 189, 1988. 389

[6] J. Camenisch. Efficient and generalized group signatures. *Advances in Cryptology - Eurocrypt '97, Lecture Notes in Computer Science 1233*, pages 465–479, 1997. 388

[7] J. Camenisch and M. Michels. Confirmer signature schemes secure against adaptive adversaries. *Advances in Cryptology - Eurocrypt 2000, Lecture Notes in Computer Science 1807*, 2000. 389

[8] D. Catalano, R. Gennaro, N. Howgrave-Graham, and P. Q. Nguyen. Paillier's Cryptosystem Revisited . *ACM CCS 2001*, 2001. 390, 393

[9] D. Chaum. Designated Confirmer Signatures. *Advances in Cryptology - Eurocrypt '94, Lecture Notes in Computer Science 950*, pages 86 – 91, 1994. 389

[10] D. Chaum and H. van Antwerpen. Undeniable signatures. *Advances in Cryptology - Crypto '89, Lecture Notes in Computer Science 435*, pages 212–216, 1990. 388

[11] D. Chaum and E. van Heyst. Group signatures. *Advances in Cryptology - Eurocrypt '91, Lecture Notes in Computer Science 547*, pages 257–265, 1991. 388

[12] R. Cramer, I. B. Damgård, and B. Schoenmakers. Proof of partial knowledge and simplified design of witness hiding protocols. *Advances in Cryptology - Crypto 94, Lecture Notes in Computer Science*, 839:174–187, 1994. 389

[13] C. Dwork, M. Naor, and A. Sahai. Concurrent Zero-Knowledge. *Proc. 30th ACM Symposium on the Theory of Computing*, pages 409 – 418, 1998. 388

[14] S. Goldwasser, S. Micali, and R. Rivest. A Secure Digital Signature Scheme. *SIAM Journal on Computing 17*, pages 281 – 308, 1988. 387

[15] M. Ito, A. Saito, and T. Nishizeki. Secret Sharing Scheme Realizing General Access Structure. *Journal of Cryptology*, 6:15–20, 1993. 390

[16] W. Jackson and K. Martin. Cumulative Arrays and Geometric Secret Sharing Schemes. *Advances in Cryptology–Auscrypt'92, Lecture Notes in Computer Science 718*, pages 48–55, 1993. 390, 397

[17] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and Their Applications. *Advances in Cryptology - Eurocrypt '96, Lecture Notes in Computer Science 1070*, pages 143 – 154, 1996. 388

[18] H. Krawczyk and T. Rabin. Chameleon hashing and signatures. *Network and Distributed System Security Symposium, The Internet Society*, pages 143 – 154, 2000. 387, 389, 390, 393, 395

[19] M. Naor. Deniable Ring Authentication. *Advances in Cryptology - Crypto 2002, Lecture Notes in Computer Science 2442*, pages 481 – 498, 2002. 386, 387, 388, 389, 400

[20] R. L. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. *Advances in Cryptology - Asiacrypt 2001, Lecture Notes in Computer Science 2248*, pages 552 – 565, 2001. 386, 387, 388

[21] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, November 1979. 390

[22] G. J. Simmons, W. A. Jackson, and K. Martin. The Geometry of Shared Secret Schemes. *Bulletin of the ICA*, 1:71 – 88, 1991. 390, 397

[23] R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. *Proceedings of Asiacrypt 2003, Lecture Notes in Computer Science*, 2003 (to appear). 389

Differential Cryptanalysis of TEA and XTEA

Seokhie Hong, Deukjo Hong, Youngdai Ko, Donghoon Chang,
Wonil Lee, and Sangjin Lee

Center for Information Security Technologies(CIST)
Korea University, Anam Dong, Sungbuk Gu, Seoul, Korea.
{hsh,hongdj,carpediem,pointchang,nice,sangjin}@cist.korea.ac.kr

Abstract. TEA and XTEA are simple block ciphers consisting of exclusive-or, addition, and shift. Although their round functions are very simple and guarantee a little security, large number of 64 rounds have made them secure enough. It seems that the best result for TEA is the related-key attack in [3], but it is less reasonable than such attacks for one key as differential and linear attacks. Impossible differential attacks on 12-round TEA and 14-round XTEA are best results except for related-key attack [5]. We suggest differential and truncated differential attacks on TEA and XTEA better than them. Our best results on TEA and XTEA are attacks on 17-round TEA and 23-round XTEA.

Keywords: Block cipher, TEA, XTEA, differential cryptanalysis, truncated differential cryptanalysis

1 Introduction

Wheeler and Needham proposed a very simple block cipher TEA in 1994 [6], which consists of exclusive-or, addition, and shift. Its round function looked too weak to give sufficient security, but TEA had large number of rounds of 64 enough to make itself secure against current attacks. Moreover, since it had a simple structure, its implementation and performance were not bad.

However, J. Kelsey, B. Schneir and D. Wagner suggested a related-key attack on full 64-round TEA in [3] at ICICS'97. Their main idea is a differential characteristic for addition mod 2^{32} with probability 1 can be constructed under a pair of keys with a particular difference. Anyway, it was an interesting result even if its assumption was not so reasonable.

In FSE 2003, D. Moon, K. Hwang, W. Lee, S. Lee, and J. Lim showed that reduced versions of TEA and XTEA are vulnerable for impossible differential attack [5] – 12-round TEA and 14-round XTEA. We want to address that Moon et al.'s work may be more important than Kelsey et al. in many cases, even if Moon et al. did not suggest an attack on full round TEA or XTEA, because an impossible differential attack is realistic than a related-key attack.

We believe that it is very interesting to establish the security TEA and XTEA under known-plaintext or chosen-plaintext attack but not related-key attack because the simplicity of TEA and XTEA is desirable for design of block ciphers. We exploit the differential properties of TEA and XTEA.

XTEA appeared as an improved version of TEA after Kelsey et al.'s related-key attack on TEA, but we show that XTEA is more vulnerable for our attacks than TEA. Our first approach is differential attack. Since TEA uses addition mod 2^{32} in combining round key with round input but XTEA uses exclusive-or, it is easier to make a differential attack on XTEA than TEA. We can construct 14-round differential characteristic of the probability of $2^{-57.79}$ for XTEA, and use it to break 16-round XTEA with 2^{61} chosen-plaintexts, while it is difficult to find a good differential characteristic for TEA.

Our second approach is a truncated differential attack. We show that TEA and XTEA have the same 8-round truncated differential characteristic of the probability of 1. However, our attack can break XTEA upto 23 rounds but TEA upto 17 rounds due to property of key schedule. This is very interesting result because 'improvement' of key schedule occupied main part of converting TEA into XTEA.

2 TEA and XTEA

TEA [6] is a 64-round Feistel block cipher with 64-bit block size and 128-bit key which consist of four 32-bit words $K[0], K[1], K[2]$, and $K[3]$. TEA does not has a key schedule algorithm. Instead, it uses the constant $\delta = 9e3779b9_x$. Let (L_n, R_n) be the input of n-th round for $1 \leq n \leq 64$. The output of n-th round is (L_{n+1}, R_{n+1}), and $L_{n+1} = R_n$. R_{n+1} is computed as follows. If $n = 2i-1$ for $1 \leq i \leq 32$,

$$R_{n+1} = L_n + (((R_n \ll 4) + K[0]) \oplus (R_n + i \cdot \delta) \oplus (R_n \gg 5 + K[1]))$$

On the other hand, if $n = 2i$ for $1 \leq i \leq 32$,

$$R_{n+1} = L_n + (((R_n \ll 4) + K[2]) \oplus (R_n + i \cdot \delta) \oplus (R_n \gg 5 + K[3]))$$

XTEA is also a 64-round Feistel block cipher with 64-bit block size like TEA. Its 128-bit secret key K is $(K[0], K[1], K[2], K[3])$, too. Using same notations as above, an input of n-th round is (L_n, R_n) and $L_{n+1} = R_n$. R_{n+1} is computed as follows. If $n = 2i-1$ for $1 \leq i \leq 32$,

$$R_{n+1} = L_n + (((R_n \ll 4 \oplus R_n \gg 5) + R_n) \oplus ((i-1) \cdot \delta + K[((i-1) \cdot \delta \gg 11) \& 3])$$

If $n = 2i$ for $1 \leq i \leq 32$,

$$R_{n+1} = L_n + (((R_n \ll 4 \oplus R_n \gg 5) + R_n) \oplus (i \cdot \delta + K[(i \cdot \delta \gg 11) \& 3])$$

3 Differential Cryptanalysis of XTEA

There is slight modification of F-function in converting TEA into XTEA. Results of both shift operations are combined by exclusive-or and then the result is added to original input of F-function by addition mod 2^{32}.

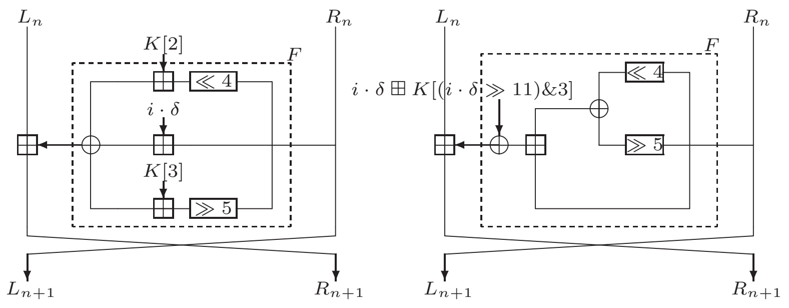


Fig. 1. $2i$-th rounds of TEA(left) and XTEA(right)

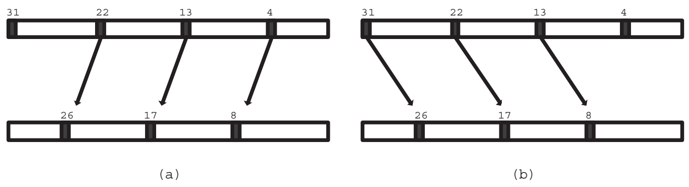


Fig. 2. Change of the XOR difference α after (a)4-bit left and (b)5-bit right shift operations

3.1 3-round Iterative Differential Characteristic of XTEA

Let $g(x)$ be $(x << 4) \oplus (x >> 5)$, the part of shift operations of F-function of XTEA. If we put a pair of inputs of F-function of the difference in which only 4, 13, 22, 31-th bits are 1, i.e $\alpha = 0x80402010$, then the output difference of g will be zero because the output differences of 4-bit left shift and 5-bit right shift are same as you see Fig. 2.

Let $\triangle_1 = x \oplus x'$ and $\triangle_2 = y \oplus y'$ where x, x', y, y' are 32-bit strings. And let $\triangle_3$ be any 32-bit string. Then let $\triangle_1 \boxplus \triangle_2 \to \triangle_3$ denote the event that the following equation holds

$$(x \boxplus y) \oplus (x' \boxplus y') = \triangle_3.$$

3-round iterative differential characteristic in Fig. 3 consists of following events.

1st round $\alpha \boxplus 0 \to \alpha;$

2nd round $\alpha \boxplus 0 \to \alpha; \;\; \alpha \boxplus 0 \to \alpha;$

3rd round $\alpha \boxplus 0 \to \alpha; \;\; \alpha \boxplus \alpha \to 0.$

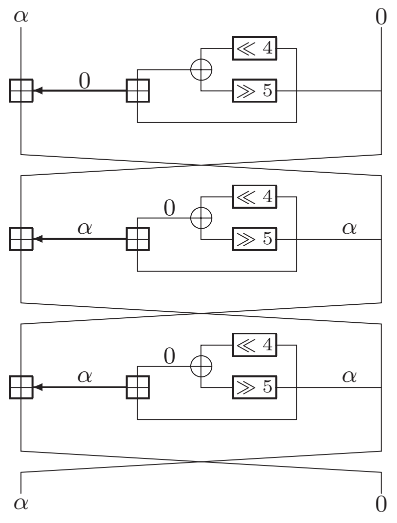


Fig. 3. 3-round iterative differential characteristic

Therefore in order to compute the probability of characteristic in Fig. 3, it is sufficient to compute the probability of the event $\alpha \boxplus 0 \to \alpha$ and $\alpha \boxplus \alpha \to 0$. We start with the following proposition which is useful in computing it. It is very easy to prove the proposition.

Proposition 1. *Let $x = (x_{31}, ..., x_0)$ and $y = (y_{31}, ..., y_0)$ be 32-bit strings. Let $z = x \boxplus y$. Let $C_i(x, y)$ for $0 \le i \le 31$ be the carry bit occurring in the i-th bit position during the addition of x and y. Then for $i = 0$ to 31, z_i can be computed by the following equations;*

$$z_i = x_i \oplus y_i \oplus C_{i-1}(x, y),$$
$$C_i(x, y) = x_i y_i \oplus x_i C_{i-1}(x, y) \oplus y_i C_{i-1}(x, y),$$

where $C_{-1}(x, y) = 0$.

Probability of $\boldsymbol{\alpha} \boxplus \mathbf{0} \to \boldsymbol{\alpha}$. Recall that the notation $\alpha \boxplus 0 \to \alpha$ means as follows. Let x and y be any 32-bit strings. And let

$$\begin{cases} z = x \boxplus y \\ z' = (x \oplus \alpha) \boxplus y \end{cases}$$

Then $\Pr(\alpha \boxplus 0 \to \alpha) = \Pr(z \oplus z' = \alpha)$. The probability is taken over uniformly random selection of x and y from $\{0, 1\}^{32}$. Note that $z \oplus z' = \alpha$ means that $z_i \oplus z'_i = \alpha_i$ for every $0 \le i \le 31$. However by the definition of the difference α, it is easy to see that if $z_i \oplus z'_i = \alpha_i$ only for $i = 5, 14$ and 23, then $z \oplus z' = \alpha$ holds.

So let us first consider the probability of $z_5 \oplus z'_5 = \alpha_5(=0)$. It is easy to see that $z_i \oplus z'_i = \alpha_i(=0)$ for $0 \le i \le 3$. Therefore $C_3(x,y) = C_3(x \oplus \alpha, y)$ holds. So the following equation holds.

$$\begin{aligned} z_5 \oplus z'_5 &= (x_5 \oplus y_5 \oplus C_4(x,y)) \oplus ((x_5 \oplus \alpha_5) \oplus y_5 \oplus C_4(x \oplus \alpha, y)) \\ &= C_4(x,y) \oplus C_4(x \oplus \alpha, y) \\ &= (x_4 y_4 \oplus x_4 C_3(x,y) \oplus y_4 C_3(x,y)) \\ &\quad \oplus (\bar{x}_4 y_4 \oplus \bar{x}_4 C_3(x \oplus \alpha, y) \oplus y_4 C_3(x \oplus \alpha, y)) \\ &= y_4 \oplus C_3(x,y), \end{aligned}$$

where $\bar{x}_4$ is a complement of x_4. As a result, the probability of $z_5 \oplus z'_5 = \alpha_5(=0)$ is $\frac{1}{2}$ because y is uniformly selected from $\{0,1\}^{32}$.

By the similar argument we can show that the probability of $z_{14} \oplus z'_{14} = \alpha_{14}(=0)$ is $\frac{1}{2}$ if $z_5 \oplus z'_5 = 0$. And we can show that the probability of $z_{23} \oplus z'_{23} = \alpha_{23}(=0)$ is $\frac{1}{2}$ if $z_{14} \oplus z'_{14} = 0$. Consequently, $\Pr(\alpha \boxplus 0 \to \alpha) = 2^{-3}$.

Probability of $\boldsymbol{\alpha \boxplus \alpha \to 0}$. It is similar with the above to compute the probability of $\alpha \boxplus \alpha \to 0$. The probability $\Pr(\alpha \boxplus \alpha \to 0)$ is also 2^{-3}.

Thus the probability of the characteristic in Fig. 3 is $(2^{-3})^5 = 2^{-15}$.

3.2 Improved 3-round Iterative Differential Characteristic of XTEA

In this section we improve the probability 2^{-15} of the characteristic in Fig. 3 to the probability $2^{-12.51}$. Note that 2nd and 3rd rounds of the characteristic in Fig. 3 have two positions in which a probability occurs respectively. See Fig 4 and 5.

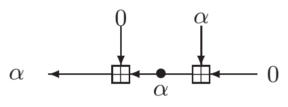


Fig. 4. 2nd round

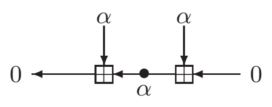


Fig. 5. 3rd round

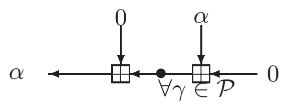


Fig. 6. 2nd round (improved prob.)

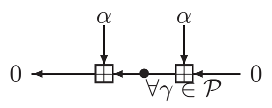


Fig. 7. 3rd round (improved prob.)

Table 1. All patterns for substring $\gamma[31-23]$, $\gamma[22-14]$, $\gamma[13-5]$, and $\gamma[4-0]$

No.	$\gamma[31-23]$	$\gamma[22-14]$	$\gamma[13-5]$	$\gamma[4-0]$
1	100000000	100000000	100000000	10000
2	100000001	100000001	100000001	·
3	100000011	100000011	100000011	·
4	100000111	100000111	100000111	·
5	100001111	100001111	100001111	·
6	100011111	100011111	100011111	·
7	100111111	100111111	100111111	·
8	101111111	101111111	101111111	·
9	111111111	111111111	111111111	·
10	011111111	011111111	011111111	·

To begin with, let us restrict our concern in the 2nd round. See Fig. 4. In Section 3.1 we considered the probability of the 2nd round as $\Pr(\alpha \boxplus 0 \rightarrow \alpha$ and $\alpha \boxplus 0 \rightarrow \alpha)$. Note that the output difference α of the first event $\alpha \boxplus 0 \rightarrow \alpha$ is just a special case of possible output differences. Now we will consider all of the possible patterns of output differences of the first event. Let $\mathcal{P}$ be the set of all of the possible patterns of output difference. Then the probability of the 2nd round can be improved by using the following formula. See Fig. 6

$$\sum_{\gamma \in \mathcal{P}} \Pr(\alpha \boxplus 0 \rightarrow \gamma \text{ and } \gamma \boxplus 0 \rightarrow \alpha)$$

The probability of the third round also can be improved by using the following formula. See Fig. 7

$$\sum_{\gamma' \in \mathcal{P}} \Pr(\alpha \boxplus 0 \rightarrow \gamma' \text{and } \gamma' \boxplus \alpha \rightarrow 0)$$

The rest of this section is devoted to show how to find all of the patterns which are elements of $\mathcal{P}$. And then we will be able to compute the probability of improved 3-round iterative characteristic Φ using the new formulas.

Let x and y be any 32-bit strings. And let $z = x \boxplus y$ and $z' = (x \oplus \alpha) \boxplus y$. Let $\gamma = z \oplus z'$. Let $\gamma[i-j]$ denote the substring from i-th bit to j-th bit of γ. Then we are now interested in finding all of the possible γ.

Recall that $\alpha = (\alpha_{31}, ..., \alpha_0)$ is the 32-bit string such that $\alpha_{31}, \alpha_{22}, \alpha_{13}$ and α_4 are all 1 and other bits are all zero. Therefore for any x and y, $\gamma_i = 0$ for $0 \leq i \leq 3$ and $\gamma_4 = 1$. But γ_5 is not uniquely determined. This fact is included in the following proposition. By proposition 2, we can consider all of the possible patterns for γ as splitting γ into $\gamma[31-23], \gamma[22-14], \gamma[13-5]$ and $\gamma[4-0]$.

Proposition 2. *For any x, y and for $j = 5, 14,$ and 23,*

1. *γ_j can be 0 or 1 with probability 2^{-1}.*
2. *For $j \leq i \leq j+6$, if γ_i is 0 then γ_{i+1} is also 0 .*

Table 2. The probabilities of each possible pattern of $\gamma[(j+8)-j]$ for $j = 5, 14, 23$

No.	$\gamma[(j+8)-j]$	Probability
1	100000000	2^{-1}
2	100000001	2^{-2}
3	100000011	2^{-3}
4	100000111	2^{-4}
5	100001111	2^{-5}
6	100011111	2^{-6}
7	100111111	2^{-7}
8	101111111	2^{-8}
9	111111111	2^{-9}
10	011111111	2^{-9}

3. *If* $\gamma_{j+7} = 0$ *then* $\gamma_{j+8} = 1$.
4. *For* $j \le i \le j+7$ *if* γ_i *is* 1 *then* γ_{i+1} *can be* 0 *or* 1 *with probability* 2^{-1}.

Proof. See the appendix.

Firstly let the set of all the possible pattern of substring $\gamma[i-j]$ be denoted as $\mathcal{P}_{[i-j]}$. Then, as you know, for any x and y there is only one pattern for $\gamma[4-0]$. So $\mathcal{P}_{[4-0]} = \{10000\}$. By proposition 2, it is easy to show $\mathcal{P}_{[13-5]} = \mathcal{P}_{[22-14]} = \mathcal{P}_{[31-23]} = \{100000000, 100000001, 100000011, 100000111, 100001111, 100011111, 100111111, 101111111, 111111111, 011111111\}$. Note that each substring $\gamma[i-j]$ is independent of other substring. (See Table 1). So $\mathcal{P}$, the set of all the possible patterns of γ, is

$$\mathcal{P} = \{s_3||s_2||s_1||s_0 : s_3 \in \mathcal{P}_{[31-23]}, s_2 \in \mathcal{P}_{[22-14]}, s_1 \in \mathcal{P}_{[13-5]}, s_0 \in \mathcal{P}_{[4-0]}\}$$

Table 2 lists the probabilities of each possible pattern of $\gamma[(j+8)-j]$ for $j = 5, 14, 23$. Then using the Table 2 we can compute the probability of 2nd round of the characteristic in Fig. 3 as follows.

$$\sum_{\gamma \in \mathcal{P}} \Pr(\alpha \boxplus 0 \to \gamma \text{ and } \gamma \boxplus 0 \to \alpha) \cong 2^{-4.755}$$

and the probability of the 3rd round of the characteristic in Fig. 3 is as follows.

$$\sum_{\gamma' \in \mathcal{P}} \Pr(\alpha \boxplus 0 \to \gamma' \text{and } \gamma' \boxplus \alpha \to 0) \cong 2^{-4.755}$$

Therefore, we can estimate the 3-round iterative characteristic of $(\alpha, 0) \to (\alpha, 0)$ more exactly as $2^{-12.51} = 2^{-3} \cdot 2^{-4.755} \cdot 2^{-4.755}$. Someone may claim that if intermediate α's are not fixed then the probability grows higher. However, you had better ignore the cases which we did not consider because fraction of them is very small.

3.3 Attack on 15-round XTEA

We can construct 15-round differential characteristic by five iterations of the 3 round characteristic. Its probability is $(2^{12.51})^5 \approx 2^{-62.55}$, so it has a little qualification to be a distinguisher for 15-round XTEA. However, we had better make an attack to find 14-th round key $K[2]$ and 15-th round key $K[3]$ of 15-round XTEA by using 13-round characteristic of the probability $2^{-54.795}$ which is formed by subtracting the first two rounds from the 15-round characteristic. It is a typical 2R-differential attack, which Biham and Shamir introduced in [1]. Since its signal-to-noise is about $2^{-9.205}$, our attack is valid for almost all keys with about 2^{59} chosen-plaintexts.

4 Truncated Differential Cryptanalysis of TEA and XTEA

TEA and XTEA have same truncated differential characteristic of the probability 1 because carry bits in addition mod 2^{32} give no effects on lower bits and TEA and XTEA use same shift operations. We show that the key schedule of XTEA makes XTEA weaker against our attack than TEA.

4.1 8-round Truncated Differential Characteristic of TEA and XTEA

In this section 8-round truncated differential characteristic of TEA and XTEA of the probability 1 is described. See Fig. 10 in Appendix B. Note that we will add one round structure of plaintexts over Ψ, so we apply Ψ from Round 2 to Round 9. There are three color of bits – white, black, gray. Any white bit of the differences denotes zero. The bit which we should observe is colored black. Its value does not change while its position is shifted upto 5 bits to the right per round. We don't care about the values of gray bits. The characteristic Ψ is valid in both TEA and XTEA because it only depends on addition and right-shift operation and because carry bits occurring in addition have effects only to the left side.

4.2 Structure of Plaintexts

We consider 1-round structure of plaintexts. We are going to construct a structure of plaintexts which contains a pair whose output difference is $(e_{30}, 0)$, so that the 8-round truncated differential characteristic Ψ can be applied from Round 2 to Round 9, and the black bit is 1.

As you see Fig. 8, when we take P_L and $P_L \oplus e_{30}$ as a pair of right halves of input of the first round of TEA, the output difference of function F is the combination of each differences of x and y. Let $\triangle_F^T, \triangle_x$, and $\triangle_y$ denote the

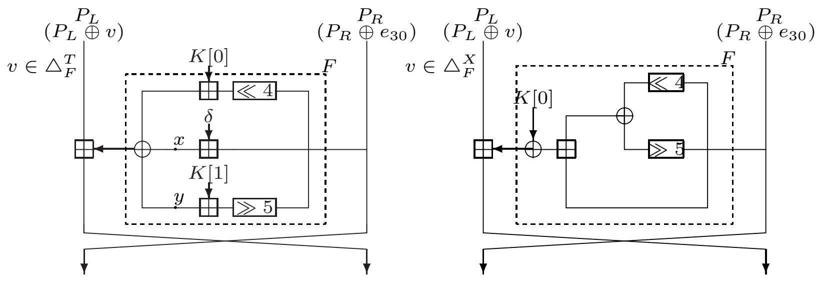


Fig. 8. Structures of plaintexts in TEA and XTEA

sets of all the possible differences of output of F-function of TEA, x, and y, respectively in Fig. 8. They are as follows.

$$\begin{aligned}
\triangle_x &= \{010\cdots0, 110\cdots0\}\\
\triangle_y &= \{00000010\cdots0, 00000110\cdots0, 00001110\cdots0, 00011110\cdots0,\\
&\quad 00111110\cdots0, 01111110\cdots0, 11111110\cdots0\}\\
\triangle_F^T &= \{w_1 \oplus w_2 : w_1 \in \triangle_x, w_2 \in \triangle_y\}
\end{aligned}$$

In order to form a structure of plaintexts, we pick a 64-bit random string $P = (P_H, P_L)$. The structure associated with P is defined as follows.

$$S_P^T = \{P\} \cup \{P \oplus (v, e_{30}) : v \in \triangle_F^T\}.$$

S_P^T contains 14 plaintext pairs of the form $(P, P \oplus (v, e_{30}))$ for $v \in \triangle_F^T$. Among the pairs, there exists exactly one pair whose output difference is $(e_{30}, 0)$ after the first round.

Similar argument is applicable to XTEA. See Fig. 8. Let $\triangle_F^X$ be the set of all the possible differences of output of F-function of XTEA. It is as follows.

$$\begin{aligned}
\triangle_F^X = \{&01000010\cdots0, 01000110\cdots0, 01001110\cdots0,\\
&01011110\cdots0, 01111110\cdots0, 00111110\cdots0,\\
&11000010\cdots0, 11000110\cdots0, 11001110\cdots0,\\
&11011110\cdots0, 11111110\cdots0, 10111110\cdots0\}
\end{aligned}$$

We form the structure S_P^X for any 64-bit plaintext P as follows.

$$S_P^X = \{P\} \cup \{P \oplus (v, e_{30}) : v \in \triangle_F^X\}.$$

S_P^X contains 12 plaintext pairs of the form $(P, P \oplus (v, e_{30}))$ for $v \in \triangle_F^X$. Among the pairs, there exists exactly one pair which derives the output difference $(e_{30}, 0)$ in the first round of XTEA.

We use structures of plaintexts and the 8-round truncated differential characteristic in Fig. 10 to attack TEA and XTEA. Now we should observe the black bit lies on 0-th bit position of the left half of the input difference of the tenth round. Note that for right key, the black bit must be 1 because input difference of the second round is $(e_{30}, 0)$. In next subsections, we focus on showing how to efficiently compute the black bit from the ciphertexts and the subkey candidates, and how many rounds allows our attacks.

4.3 Attack on 17-round TEA

Fig. 11 in the appendix depicts how to compute the black bit from the ciphertexts, in which we should compute the difference of every dotted bit position and the values of a bit pair of every gray bit position in order to get the black bit of the left half of the input difference of the tenth round.

For example, if we know the differences 0-th bits of right halves and 0-th and 5-th bits of left halves of output pair of the tenth round, then we can compute the black bit of input difference of tenth round. In the eleventh round, we need $0 \sim 10$-th bits of left halves and $0 \sim 5$-th bits of right halves of output pair of eleventh round and $0 \sim 4$-th bits of $K[0]$ and $K[1]$ in order to compute the gray bits of the input pair of the eleventh round – actually, for 10-th bits of left halves and 5-th bits of right halves of output pair of eleventh round we may get only differences. The key bits related to the black bit increase by 5 bits every round. Consequently, if we guess the whole of $K[0]$ and $K[1]$, and $0 \sim 30$-th bits of $K[2]$ and $K[3]$, and get the output pair of 17-th round, we can compute the black bit of input difference of the tenth round.

The algorithm performing our truncated differential attack on 17-round TEA is depicted in Fig. 9. Let K_j denote guessed values of $K[0]||K[1]$, respectively and let $K's$ denote the concatenation of guessed values of $0 \sim 29$-th bits of $K[2]$ and $K[3]$, respectively. Note that we find 124 bits of $K = (K[0], K[1], K[2], K[3])$.

The output of the algorithm is right value of 124 bits of $K = (K[0], K[1], K[2], K[3])$ with high probability if m is sufficiently large. The probability that the attack algorithm outputs i-th key-candidate for $0 \leq i \leq 2^{124}-1$ is $(1-2^{-m})^i$. So, the average of the success rate of the attack on 17-round TEA is

$$2^{-124} \sum_{i=0}^{2^{124}-1} (1-2^{-m})^i = 2^{m-124}(1-(1-2^{-m})^{2^{124}}) \approx 2^{m-124}(1-e^{-2^{124-m}}).$$

For m structures of plaintexts, the expected number of trials required until each key-candidate is determined as a wrong value is $1 + 2^{-1} + 2^{-2} + \cdots + 2^{-m+1} = 2 - 2^{-m+1}$. If a key-candidate is right, then the number of trials is exactly m. Then the average number of trials of the attack is

$$2^{-k} \sum_{i=0}^{2^k-1} (m + i \cdot (2 - 2^{-m+1})) = m + (1-2^{-m})(2^k - 1).$$

In the attack algorithm, one trial is almost equal to one 11-round TEA encryption. So, if we get 128 structures of plaintexts, the attack on 17-round TEA

```
Input : $m$ − structures : $S^T_{P^1}, S^T_{P^2}, \cdots, S^T_{P^m}$ and their ciphertexts
Output : 124 bit partial key values of $K = (K[0], K[1], K[2], K[3])$

1. For $K_j$ $(j = 1, \cdots, 2^{64})$
  1.1. For each $i = 1, \cdots, m,$ find $v_{j_i}$ such that
       $(P^i_L + F(P^i_R, K_j)) \oplus ((P^i_L \oplus v_{j_i}) + F(P^i_R \oplus e_{30}, K_j)) = 0$
       / * For convenience' sake, let $C_i$ be the ciphertext of $P^i$ and $C^*_i$ be that
         of $P^i \oplus (v_{j_i}, e_{30})$ * /
  1.2. For $K'_s$ $(s = 1, \cdots, 2^{60})$
    1.2.1. For $i = 1, \cdots, m,$ compute $\sigma_i = \sigma(C_i, C^*_i, K_j, K'_s)$
           if($\sigma_i = 1$ and $i = m$) output $K_j, K'_s$and terminate
           else if $(\sigma_i = 0)$ goto 1.2
```

Fig. 9. Algorithm of the truncated differential attack on 17-round TEA

will succeed on average with probability 96.9%. This success rate implies our attack reduces almost all key spaces efficiently. It has data complexity of $128 \cdot 15 = 1920$ chosen-plaintexts and time complexity of $(128+(1-2^{-128})(2^{124}-1)) \cdot \frac{5.5}{17} \cdot 2 \approx 2^{123.37}$ 17-round TEA encryptions.

4.4 Attacks on Reduced Rounds of XTEA

The key schedule of XTEA is slightly irregular unlike TEA. We find XTEA is more breakable than TEA due to this property. The algorithm of any truncated differential attack on XTEA is similar to that of TEA.

According to Table 4, from 24-*th* round to 30-*th* round, $K[3]$ is not used. This means that "free 5 rounds" can be appended to Fig. 11. So 22-round XTEA can be broken by our truncated differential attack. For success rate 96.9%, this attack requires 1625 chosen-plaintexts and $2^{120.71}$ 22-round XTEA encryptions to recover 121 bits of $K = (K[0], K[1], K[2], K[3])$. We can consider attacks on any other reduced rounds of XTEA. For example, if we regard Round 9 $\sim$ Round 30 of XTEA as a variant of XTEA, our truncated differential attack breaks it with similar complexity.

We can extend the attacks on XTEA by exploiting the structures of plaintexts. Let a plaintext P be chosen at random and let A be the set of all 32-bit values whose lower 21-bit is fixed as $10 \cdots 0$. We define 2-round structure of plaintexts S^{XX}_P as follows.

$$S^{XX}_P = \{P\} \cup \{P \oplus (w, v) : w \in A, v \in \triangle^X_F\}$$

S^{XX}_P contains 12,289 chosen-plaintexts and 12,288 plaintext pairs of the form $(P, P \oplus (w, v))$ where $w \in A$ and $v \in \triangle^X_F$. Among the pairs, there exists at least

Table 3. Attacks on reduced rounds of XTEA

Rounds	Key Bits
21 (Round 1 $\sim$ Round 21)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(32 bits), $K[3]$(25 bits)
22 (Round 9 $\sim$ Round 30)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(32 bits), $K[3]$(25 bits)
21 (Round 18 $\sim$ Round 38)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(32 bits), $K[3]$(25 bits)
21 (Round 23 $\sim$ Round 43)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(25 bits), $K[3]$(32 bits)
22 (Round 31 $\sim$ Round 52)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(25 bits), $K[3]$(32 bits)
21 (Round 36 $\sim$ Round 56)	$K[0]$(25 bits), $K[1]$(32 bits), $K[2]$(32 bits), $K[3]$(32 bits)
21 (Round 41 $\sim$ Round 61)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(32 bits), $K[3]$(25 bits)
23 (Round 8 $\sim$ Round 30)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(32 bits), $K[3]$(25 bits)
22 (Round 22 $\sim$ Round 43)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(25 bits), $K[3]$(32 bits)
23 (Round 30 $\sim$ Round 52)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(25 bits), $K[3]$(32 bits)
22 (Round 40 $\sim$ Round 61)	$K[0]$(32 bits), $K[1]$(32 bits), $K[2]$(32 bits), $K[3]$(25 bits)

Table 4. Key schedule of XTEA

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Key	$K[0]$	$K[3]$	$K[1]$	$K[2]$	$K[2]$	$K[1]$	$K[3]$	$K[0]$	$K[0]$	$K[0]$	$K[1]$	$K[3]$	$K[2]$	$K[2]$	$K[3]$	$K[1]$
Round	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Key	$K[0]$	$K[0]$	$K[1]$	$K[0]$	$K[2]$	$K[3]$	$K[3]$	$K[2]$	$K[0]$	$K[1]$	$K[1]$	$K[1]$	$K[2]$	$K[0]$	$K[3]$	$K[3]$
Round	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
Key	$K[0]$	$K[2]$	$K[1]$	$K[1]$	$K[2]$	$K[1]$	$K[3]$	$K[0]$	$K[0]$	$K[3]$	$K[1]$	$K[2]$	$K[2]$	$K[1]$	$K[3]$	$K[1]$
Round	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Key	$K[0]$	$K[0]$	$K[1]$	$K[3]$	$K[2]$	$K[2]$	$K[3]$	$K[2]$	$K[0]$	$K[1]$	$K[1]$	$K[0]$	$K[2]$	$K[3]$	$K[3]$	$K[2]$

one pair whose output difference is $(e_{30}, 0)$ for each $K[0], K[1]$ after two rounds. For the expectation 96.9% of success rate, we use 125 structures to make an attack on 23-round XTEA which requires $125 \cdot 12,289 \approx 2^{20.55}$ chosen-plaintexts and $(125 + (1 - 2^{-125})(2^{121} - 1)) \cdot \frac{9}{23} \cdot 2 \approx 2^{120.65}$ 23-round XTEA encryptions. (e.g., see Fig. 12)

We list attacks on more than 21 rounds of XTEA in Table 3. 'Rounds' denotes the number and the range of rounds broken. 'Key Bits' denotes bit-length of $K[0], K[1], K[2]$, and $K[3]$ recovered by the attack. First 7 attacks use 1-round structure S_P^X and the others use 2-round structure S_P^{XX}.

5 Conclusion

We explained differential and truncated differential attacks on TEA and XTEA in this paper. We found the 3-round iterative differential characteristic of XTEA of the probability $2^{-12.51}$ from which we constructed 13-round and 15-round characteristics of the probability $2^{-54.795}$ and $2^{-62.55}$ respectively, and we implied that the later characteristic can be used to break 15-round XTEA with about 2^{59} chosen-plaintexts. We also found the truncated differential characteristic of the probability 1 which holds on both TEA and XTEA. Our best results are the truncated differential attacks on 17-round TEA and 23-round XTEA.

The attack on 17-round TEA requires 1920 chosen-plaintexts and $2^{123.37}$ encryption times. The attack on 23-round XTEA requires $2^{20.55}$ chosen-plaintexts and $2^{120.65}$ encryption times. We also computed success rates of our attack algorithms, which imply key spaces are reduced rapidly. It is clear that our results are much better than [5] of FSE 2002.

We want to address that it is interesting to study the security of TEA and XTEA because no complex theories are not applied to their design strategy. We believe that better attacks will appear than ours.

References

[1] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems", *CRYPTO'90*, LNCS 537, Springer-Verlag, 1991, pp.2–21. 409

[2] D. Hong, Y. Ko, D. Chang, W. Lee, and J. Lim, "Differential Cryptanalysis of XTEA", Available on http://cist.korea.ac.kr/Tr/TR03-13.ps.

[3] J. Kelsey, B. Schneir and D. Wagner, "Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA", *ICICS'97*, LNCS 1334, Springer-Verlag, 1997, pp. 203–207. 402

[4] L. R. Knudsen, "Truncated and Higher Order Differential", *FSE'94*, LNCS 1008, pp.229–236, Springer-Verlag, 1995.

[5] D. Moon, K. Hwang, W. Lee, S. Lee, and J. Lim, "Impossible Differential Cryptanalysis of Reduced Round XTEA and TEA", *FSE 2002*, LNCS 2365, Springer-Verlag, 2002, pp. 49–60. 402, 414

[6] D. J. Wheeler and R. M. Needham, "TEA, a Tiny Encryption Algorithm", *FSE'94*, LNCS 1008, Springer-Verlag, 1994, pp. 97–110. 402, 403

[7] R. M. Needham and D. J. Wheeler, "eXtended Tiny Encryption Algorithm", October, 1997, Available on http://vader.brad.ac.uk/tea/tea.shtml.

A Proof of Proposition 2

Let $j = 5$. We don't present the proof for $j = 14$ and 23 because it is very similar to the case of $j = 5$.

1. Since $\gamma_5 = z_5 \oplus z'_5 = y_4 \oplus C_3(x, y)$ and y is uniformly selected from $\{0,1\}^{32}$, the argument holds.
2. For $5 \leq i \leq 11$, note that $\gamma_i = C_{i-1}(x, y) \oplus C_{i-1}(x \oplus \alpha, y)$. Then by the assumption $\gamma_i = 0$,

$$\begin{aligned}\gamma_{i+1} = z_{i+1} \oplus z'_{i+1} &= (x_{i+1} \oplus y_{i+1} \oplus C_i(x,y)) \oplus (x_{i+1} \oplus y_{i+1} \oplus C_i(x \oplus \alpha, y)) \\ &= C_i(x,y) \oplus C_i(x \oplus \alpha, y) \\ &= (x_i y_i \oplus x_i C_{i-1}(x,y) \oplus y_i C_{i-1}(x,y)) \\ &\quad \oplus (x_i y_i \oplus x_i C_{i-1}(x \oplus \alpha, y) \oplus y_i C_{i-1}(x \oplus \alpha, y)) \\ &= (x_i \oplus y_i)(C_{i-1}(x,y) \oplus C_{i-1}(x \oplus \alpha, y)) \\ &= (x_i \oplus y_i)\gamma_i \\ &= 0.\end{aligned}$$

3. Since γ_{12} is 0,

$$\begin{aligned}\gamma_{13} = z_{13} \oplus z'_{13} &= \alpha_{13} \oplus C_{12}(x,y) \oplus C_{12}(x \oplus \alpha, y) \\ &= 1 \oplus (x_{12} \oplus y_{12})(C_{11}(x,y) \oplus C_{11}(x \oplus \alpha, y)) \\ &= 1 \oplus (x_{12} \oplus y_{12})\gamma_{12} = 1.\end{aligned}$$

4. By the equations in the proof of 2 and 3 and the fact that x and y are uniformly selected from $\{0,1\}^{32}$, it is easy to prove the argument.

□

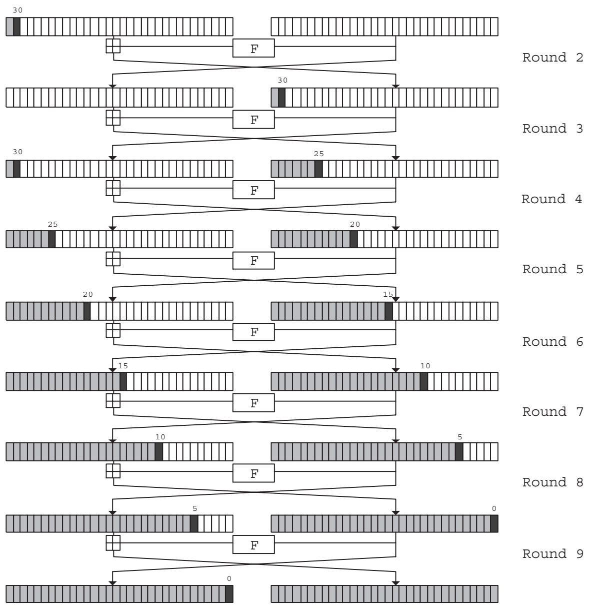


Fig. 10. 8-round truncated differential characteristic of the probability 1

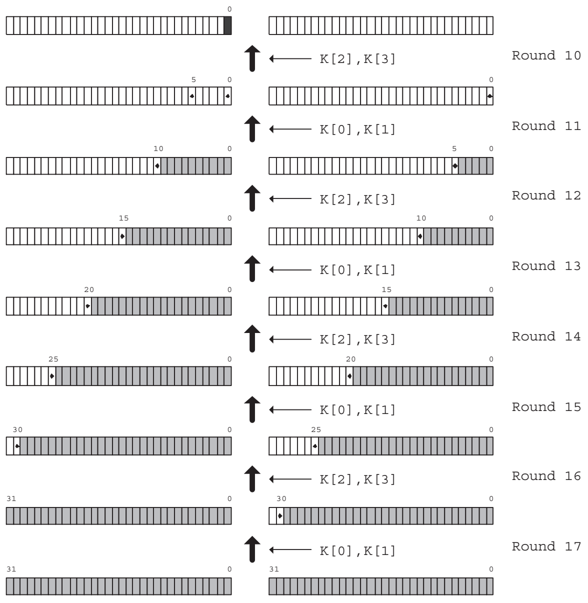


Fig. 11. Attack on 17-round TEA. This is also applied to 17-round XTEA with the key schedule of XTEA

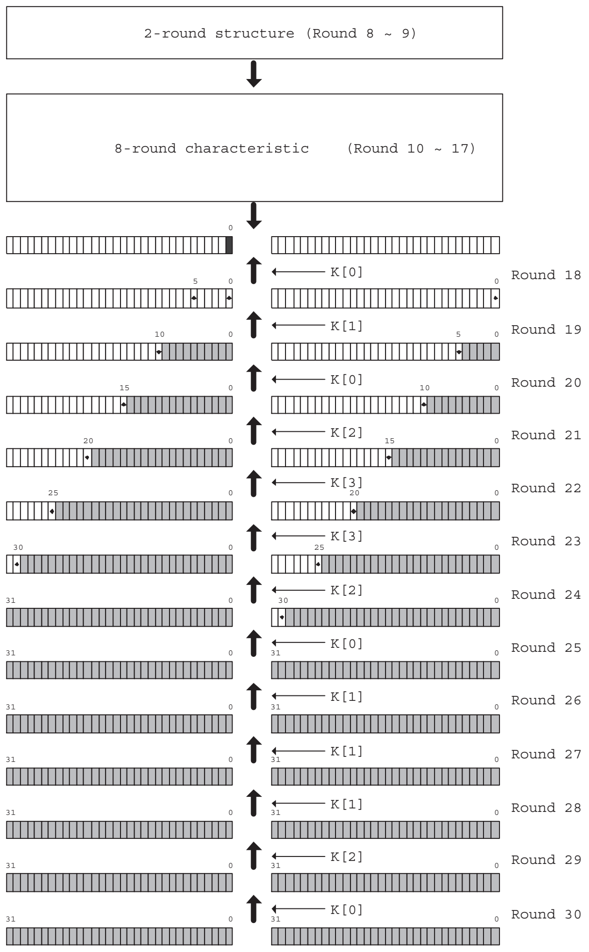


Fig. 12. Attack on 23-round XTEA (Round 8 ∼ Round 30)

A Complete Divide and Conquer Attack on the Alpha1 Stream Cipher*

K. Chen, L. Simpson, M. Henricksen, W. Millan, and E. Dawson

Information Security Research Centre,
Queensland University of Technology,
GPO Box 2434, Brisbane, Queensland, 4001, Australia.
{k.chen,lr.simpson,m.henricksen,b.millan,e.dawson}@qut.edu.au

Abstract. Alpha1 is a stream cipher with a key size of 128 bits. It was proposed as a replacement algorithm for the stream cipher A5 to supply confidentiality over mobile communication systems. Alpha1 consists of four binary linear feedback shift registers. Previous attacks on Alpha1 only recover the initial state of the shortest register. In this paper we present a complete divide and conquer attack, starting with an improved attack on the shortest register, and continuing the attack to recover the remaining three registers. Although Alpha1 is a 128-bit stream cipher, the key can be recovered through this divide and conquer attack with complexity 2^{61}, using 35,000 bits of known plaintext.

Keywords. Cryptanalysis, stream cipher, Alpha1, irregular clocking, divide and conquer attack

1 Introduction

Alpha1 [6] was proposed in 2001 as a stream cipher algorithm for mobile and wireless devices. Alpha1 follows the design of the A5 [1] closely and is potentially a replacement for this stream cipher. Both ciphers are based on linear feedback shift registers (LFSRs) and use irregular clocking. A5 is based on three irregularly clocked LFSRs,with a total key size of 64 bits whereas Alpha1 uses four LFSRs, with a total key size of 128 bits. In Alpha1, one of the four registers is regularly clocked while the other three registers are irregularly clocked. The contents of two stages from each of the irregularly clocked registers control the clocking according to a majority function, so that at least one of these three registers is clocked for each output bit.

Several weaknesses and inaccuracies in the Alpha1 specification are noted in [8], which also outlines a theoretical attack to recover the initial state of the shortest (29 bit) register of Alpha1. In [11], a similar approach is used in an attack which recovers the initial state of this register. In both cases, the recovery of the initial states of the other three registers (totalling 99 bits) is left as an open problem. This paper addresses that problem. We present a complete divide

* This research is partially sponsored by MSRC, Kyungpook National University

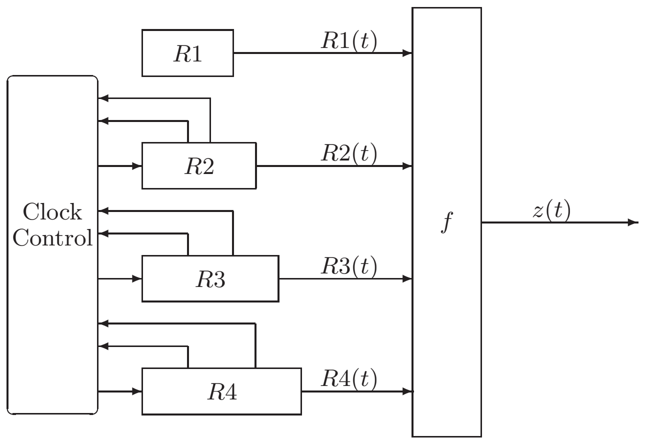


Fig. 1. Alpha1 Stream Cipher

and conquer attack on Alpha1, which begins with an improvement of the attack in [11], and continues the divide and conquer approach to recover the initial states of the remaining three registers, using a correlation measure based on edit distances, similar to that used in attacks on alternating step generator [3], bilateral stop/go generator [7], and A5-type generators [4].

This paper is organised as follows: Section 2 contains a description of Alpha1. Section 3 outlines weaknesses and previous attacks. Section 4 describes our divide and conquer attack on Alpha1. Finally, Section 5 contains some concluding remarks on the security provided by Alpha1.

2 Description of Alpha1

Alpha1 uses four binary LFSRs of lengths 29, 31, 33 and 35, respectively. These registers are denoted $R1$, $R2$, $R3$, and $R4$, as shown in Figure 1. The LFSRs have the following feedback polynomials:

$$\begin{aligned} f_1(x) &= x^{29} + x^{27} + x^{24} + x^8 + 1 \\ f_2(x) &= x^{31} + x^{28} + x^{23} + x^{18} + 1 \\ f_3(x) &= x^{33} + x^{28} + x^{24} + x^4 + 1 \\ f_4(x) &= x^{35} + x^{30} + x^{22} + x^{11} + x^6 + 1 \end{aligned}$$

At time t, denote the output of register Ri as $Ri(t)$ and the output keystream of Alpha1 as $z(t)$. The keystream bit is a function of the output bit of each of the four registers.

$$\begin{aligned} z(t) &= f(R1(t), R2(t), R3(t), R4(t)) \\ &= R1(t) \oplus R2(t) \oplus R3(t) \oplus R4(t) \oplus (R2(t) \text{ AND } R3(t)) \\ &= R1(t) \oplus R4(t) \oplus (R2(t) \text{ OR } R3(t)) \end{aligned}$$

Let Ri_j denote the jth stage of LFSR i. $R1$ is regularly clocked and the other three LFSRs are irregularly clocked in a stop/go fashion (the output bits of the LFSRs occur one or more times in the LFSR output sequence). Each of $R2$, $R3$ and $R4$ has two clocking taps. These six clocking taps are divided into two groups, each containing one tap from each of the three irregularly clocked LFSRs:

$$\text{Group 1} : R2_{10}, R3_{22}, R4_{11}$$
$$\text{Group 2} : R2_{21}, R3_{10}, R4_{24}$$

For each group, a majority bit is calculated. For group 1, the majority bit is calculated as $\text{Maj}_1(R2, R3, R4) = (R2_{10} + R3_{22} + R4_{11}) \gg 1$, where $\gg$ denotes the right shift operation. For group 2, the majority bit is calculated as $\text{Maj}_2(R2, R3, R4) = (R2_{21} + R3_{10} + R4_{24}) \gg 1$.

Register Ri $(i = 2, 3, 4)$ is clocked when both of its clocking taps agree with the majority bit of the respective groups. For example, $R2$ is clocked only if the content of $R2_{10}$ agrees with the majority bit of group 1 and the content of $R2_{21}$ agrees with the majority bit of group 2. For each keystream bit produced, between two and four of the Alpha1 registers are clocked, including $R1$ which is always clocked.

3 Known Weaknesses and Previous Analysis

In [8], Mitchell notes two weaknesses and an inaccuracy in the specification of Alpha1. Firstly, the feedback polynomial $f_4(x)$ is not primitive. Secondly, the combining function f can be approximated by a linear function. Also the authors of Alpha1 miscalculated the clocking probability.

The feedback polynomial $f_4(x)$ is not primitive,which can be seen by the even number of terms. This polynomial can be factorised into four smaller degree polynomials, one of degree 22, one of degree 11, and two of degree 1. As $f_4(x)$ is not primitive, the sequence it generates is not of maximal length. For example, if $R4$ is initialised with all ones, then the $R4$ output sequence will be all ones. Note that this implies Alpha1 has at least 2^{93} weak keys, because if $R4$ is initialised with all ones, the contribution from $R4$ to both the clocking and the combining function is constant, and the Alpha1 output keystream depends only on $R1$, $R2$ and $R3$.

The authors of Alpha1 claimed that $R2$, $R3$ and $R4$ will each be clocked with a probability of 7/13. However, in [8] it is noted that if the inputs to the six clocking taps are randomly distributed, then $R2$, $R3$ and $R4$ will each be clocked with a probability of 9/16.

The combining function $f(R1(t), R2(t), R3(t), R4(t))$ is closely approximated by a linear combination of the output from registers $R1$ and $R4$. The output function $z(t) = R1(t) \oplus R4(t) \oplus 1$ holds with probability equal to 0.75. The correct clocking probability and this bias in the combining function can be exploited in the known-plaintext attack outlined in [8] to recover the initial state of $R1$.

The attack involves exhaustively searching through all 2^{29} initial states of $R1$, producing a length of candidate keystream from which a correlation measure is calculated. Note that this attack was not implemented, therefore the complexity of testing each guess is unknown. If k denotes the complexity of testing each $R1$ candidate state, then the complexity of this attack on $R1$ is $2^{29} \cdot k$. Once the initial state of $R1$ is recovered, this leaves the initial states of the other three registers (99 bits) to be obtained through exhaustive search. Thus applying this attack to recover the initial states of all four registers has the complexity $2^{29} \cdot k + 2^{99}$.

In [11], it is shown that the clocking taps are not randomly distributed and are in fact biased. An attack to recover the initial state of $R1$ uses a similar method to that described in [8]. The keystream can be rewritten as the regularly clocked $R1$ output sequence plus the sub-keystream of the remaining registers $z'(t) = R1(t) \oplus z(t)$. By guessing the initial state of $R1$, the corresponding $z'(t)$ can be produced. Using the biased clocking, for the correct $R1$ initial state the distribution of digraphs 00, 01, 10, 11 in $z'(t)$ is $\frac{19}{64}$, $\frac{13}{64}$, $\frac{13}{64}$, and $\frac{19}{64}$, respectively. If the distribution of digraphs in $z'(t)$ agrees with the calculated values above, the guessed initial state of $R1$ is deemed correct. It is claimed that this attack has complexity 2^{29} and requires about 3000 bits of known keystream for a success rate of about 90%. Note that the estimated complexity of 2^{29} ignores the work required to test each candidate. As the algorithm iterates through 3000 bits of keystream for each of the 2^{29} possible $R1$ initial states, we estimate a complexity of $2^{29} \cdot 3000 \approx 2^{40.6}$ operations. We express the complexity in terms of operations to permit a comparison with our attack presented in Section 4.

4 Divide and Conquer Attack

In this section, we outline our divide and conquer attack against Alpha1 that sequentially recovers the initial states of all four registers. Firstly, the initial state of $R1$ is recovered using a slightly improved version of the attacked presented in [11]. Secondly, we recover the initial state of $R4$ using a probabilistic correlation attack. Thirdly, we exhaustively search through the initial states of $R2$, and reconstruct the initial state of $R3$.

4.1 Recovery of *R*1

In [11], the initial state of $R1$ is recovered by calculating the distribution of digraphs in $z'(t)$ for each candidate $R1$ initial state, and comparing that with the expected distribution for the correct initial state. Note that the digraphs can be divided into two groups, one with identical bit values (00 and 11) and the other with unlike values (01 and 10). For the correct initial state of $R1$, the distribution is symmetric: both 00 and 11 occur with the same probability ($\frac{19}{64}$) and 01 and 10 both occur with probability $\frac{13}{64}$. We can combine these probabilities and detect the correct initial state of $R1$ by calculating the binary derivative $d'(t)$ of $z'(t)$, where $d'(t) = z'(t-1) \oplus z'(t)$ for $t \geq 1$. This is similar to the approach taken by Mitchell in [8]. The binary derivative measures the number of bits in a binary sequence that differ from the previous bit value. The digraphs with identical bit values will have binary derivative of 0, and the others have binary derivative of 1. Based on the probabilities calculated in [11], the bias in the binary derivative is derived for the correct $R1$ initial state. The distribution of zeros and ones in the binary derivative for the correct state is $\frac{38}{64}$ and $\frac{26}{64}$, respectively. For an incorrectly guessed initial state, there is no discernable bias.

Assuming we observe some keystream $\{z(t)\}_{t=0}^{n-1}$ of Alpha1, we use the following algorithm to recover the initial state of Alpha1:
Inputs: Alpha1 keystream $\{z(t)\}_{t=0}^{n-1}$, threshold value c
Output: initial register state of $R1$

1. Guess $\hat{R}1$, a candidate initial state for $R1$, and produce the $\hat{R}1$ output sequence $\{\hat{R1}(t)\}_{t=0}^{n-1}$.
2. Calculate $\{z'(t)\}_{t=0}^{n-1}$ by adding $\{\hat{R1}(t)\}_{t=0}^{n-1}$ to $\{z(t)\}_{t=0}^{n-1}$.
3. Calculate the binary derivative $\{d'(t)\}_{t=1}^{n-1}$ of $\{z'(t)\}_{t=0}^{n-1}$.
4. Calculate the proportion of $\{d'(t)\}_{t=1}^{n-1}$ with the value 1. If the proportion is less than $\frac{26}{64} + c$ (where c is a one-sided threshold based upon keystream length), then put the guessed $\hat{R}1$ state into the list of candidate initial states.

This attack requires exhaustive search over all 2^{29} states of $R1$ and the production of n bits of output keystream for each state. The attack therefore has complexity $2^{29} \cdot n$.

Experimental Results. To efficiently recover $R1$, optimal values for threshold value c and keystream length n have to be determined. If the threshold value is too small, then the correct candidate may be inadvertently discarded; if it is too high, then a large number of false positives may result.

We experimented with five different threshold values $\frac{2}{128} \leq c \leq \frac{6}{128}$ (in steps of $\frac{1}{128}$) and six keystream lengths $512 \leq n \leq 5120$. The attack was run 2^{16} times for each threshold value and keystream length.

The results of the experiment are shown in Table 1. The proportion of cases for which the true initial state was successfully identified and the average number of false positives are recorded in columns 3 and 4 respectively. For example, for $c = \frac{3}{128}$ and $n = 2048$, the attack successfully identified the correct initial state in 96.8% of cases with around 8 false positives. Increasing the length of keystream to $n = 3072$ found correct initial states in 98.8% of cases, with no false positives being identified. Generalizing this, an increase in the keystream length allows a corresponding decrease in the threshold without generating extra false positives. However, the keystream length is linearly related to the time that the attack requires to determine the correct initial state.

For the parameters $c = \frac{3}{128}$ and $n = 2048$, it takes around one hour to run the attack on a Pentium 4 1.8GHz desktop with a success rate of 96.8%. The complexity of the attack is $2^{29} \cdot n = 2^{40}$ for $n = 2048$. In comparison, the attack presented in [11] took several hours on a Pentium 4, with a success rate of 0.90 for $n = 3000$. Note that the attack described in [11] calculates the distribution

Table 1. Table of Success Rates

threshold c	keystream length n	success rate	false positives
$\frac{2}{128}$	512	0.750	2^{17}
	768	0.788	2^{12}
	1024	0.816	$2^{8.0}$
	2048	0.895	0
	3072	0.935	0
	4096	0.960	0
$\frac{3}{128}$	512	0.840	2^{19}
	768	0.878	2^{15}
	1024	0.910	2^{11}
	2048	0.968	$2^{3.0}$
	3072	0.988	0
	4096	0.995	0
$\frac{4}{128}$	512	0.900	2^{20}
	768	0.938	2^{17}
	1024	0.961	2^{14}
	2048	0.993	$2^{4.0}$
	3072	0.999	0
	4096	1.000	0
$\frac{5}{128}$	512	0.944	2^{22}
	768	0.973	2^{19}
	1024	0.986	2^{17}
	2048	0.999	$2^{8.1}$
	3072	1.000	1
	4096	1.000	0
$\frac{6}{128}$	512	0.971	2^{23}
	768	0.989	2^{21}
	1024	0.996	2^{20}
	2048	1.000	2^{13}
	3072	1.000	$2^{5.8}$
	4096	1.000	$2^{4.5}$

across four digraphs for every initial $R1$ state. Our attack on $R1$ only needs to calculate a distribution with two states, and is therefore more efficient than [11].

The running time of our attack can be further reduced by running a two stage process. First determine a pool of candidate initial states using $n = 1024$ and $c = \frac{4}{128}$, and then search that pool after increasing n to 2048 and decreasing c to $\frac{3}{128}$. This attack takes 40 minutes, with a complexity of just over 2^{39} and a success rate of 93%.

4.2 Reduced Version of Alpha1

As the initial state of $R1$ has already been recovered in Section 4.1, in the remainder of this Section we consider a *reduced* version of Alpha1 with $R1$ removed, as shown in Figure 2. This reduced Alpha1 is similar to A5 as it consists of

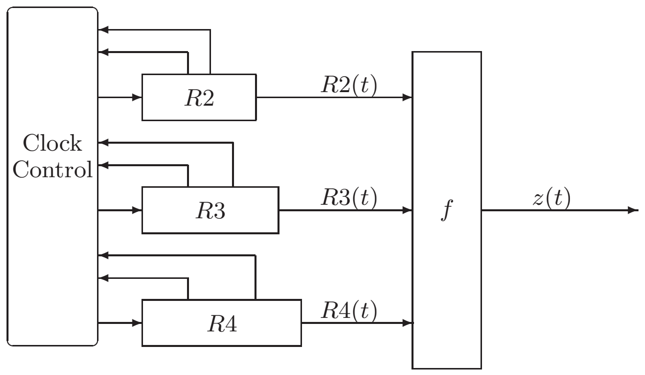


Fig. 2. Reduced Version of Alpha1 ($R1$ Removed)

Table 2. Truth Table for f' in Reduced Version of Alpha1

$R2(t)$	$R3(t)$	$R4(t)$	$z'(t)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

three irregularly clocked LFSRs, although the clocking mechanism is more complex than that os A5. The clocking mechanism is similar to that of the bilateral stop/go generator [12, 13].

Denote the keystream for reduced Alpha1 as $z'(t)$. As noted in Section 3, this can be obtained from the keystream of Alpha1 by adding the regularly clocked $R1$ output sequence $\{R1(t)\}_{t=0}^{n-1}$ to the observed keystream $\{z(t)\}_{t=0}^{n-1}$. Let f' define the combining function for the reduced Alpha1. The output keystream at time t can be written as $z'(t) = f'(R2(t), R3(t), R4(t)) = R4(t) \oplus (R2(t) \text{ OR } R3(t))$.

The clocking mechanism for registers R2, R3 and R4 forming the reduced version of Alpha1 is identical to that of the original Alpha1, described in Section 2.

From Table 2, note that $z'(t)$, the output of the reduced version of Alpha1, is strongly correlated to the input value $R4(t)$. $R4(t)$ disagrees with $z'(t)$ in 6 cases out of 8, so $P(z'(t) \neq R4(t)) = 0.75$. This bias enables a correlation attack targeting $R4$, provided an appropriate measure of correlation can be found. The correlation measure used in our attack on $R4$ is the joint probability, and is described in the next Section.

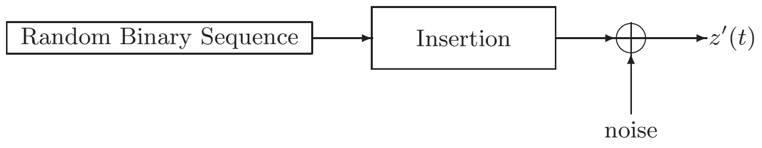


Fig. 3. Model for the Probabilistic Attack

4.3 Recovery of *R*4

The classical correlation attack described in [9], using correlation measures based upon Hamming distance, is useful when the keystream sequence and underlying LFSR sequence are of identical length. Irregular clocking of LFSRs, as occurs for Alpha1, produces a keystream sequence that has different length from the underlying LFSR sequence. This provides resistance to the classical correlation attack. However, a correlation attack can still be performed, but a different correlation measure is required.

Model for the Probabilistic Attack. For the correlation attack on $R4$, view the keystream of the reduced Alpha1, $z'(t)$, as a version of the underlying $R4$ sequence, to which bit insertion and additive noise have been applied, as shown in Figure 3. Bit insertion allows for the increased length of the keystream segment compared to the regularly clocked $R4$ sequence (due to the stop/go clocking) and the additive noise allows for the contribution from $R2$ and $R3$. The task of the cryptanalyst is to determine the correct initial state of $R4$ given a segment of the keystream $\{z'(t)\}_{t=0}^{n-1}$. A possible correlation measure is the joint probability as proposed in [5]. This is based upon Levenshtein distances, and has been successfully used in attacking another irregularly clocked keystream generator, the shrinking generator [2] as described in [10]. Similar approaches have resulted in the successful correlation attacks on other irregularly clocked generators in the literature, including alternating step generator [3], bilateral stop/go generator [7], and A5-type generators [4]. The Levenshtein distance is the minimum number of edit operations needed to transform one sequence to another. Valid edit operations include bit insertion, bit deletion and bit complementation. To attack $R4$ of the reduced Alpha1, where stop/go clocking has been used, bit insertion and bit complementation are required.

Joint Probability The joint probability $P(A^m, B^n)$ for arbitrary binary input and output strings $A^m = \{a(t)\}_{t=1}^{m}$ and $B^n = \{b(t)\}_{t=1}^{n}$, respectively, is computed using a recursive algorithm [5] based on string prefixes. Let $A^s = \{a(t)\}_{t=1}^{s}$ denote the prefix of A^m of length s and $B^{e+s} = \{b(t)\}_{t=1}^{e+s}$ denote the prefix of B^n of length $e+s$. Let $P(s,e)$ denote the partial joint probability for A^s and B^{e+s}, for $1 \leq s \leq m$ and $0 \leq e \leq n-m$. Let $\delta(a,b)$ denote the complementation probability, with $\delta = (1 - \text{noise probability})$ if $a = b$

Table 3. Table of Joint Probabilities

p=0.4375	100		200		300	
%	RAND	CORR	RAND	CORR	RAND	CORR
0	0	1.24E-26	0	4.81E-54	0	2.07E-82
5	0	1.01E-23	0	5.45E-49	0	1.54E-75
10	0	3.45E-23	0	2.96E-48	0	1.23E-74
15	0	8.75E-23	0	8.68E-48	0	4.10E-74
20	0	1.66E-22	0	2.15E-47	0	1.21E-73
25	0	2.90E-22	0	4.46E-47	0	2.98E-73
30	0	4.81E-22	0	8.59E-47	0	6.91E-73
35	3.05E-31	7.58E-22	0	1.62E-46	0	1.47E-72
40	2.94E-29	1.19E-21	0	2.90E-46	0	3.23E-72
45	2.12E-28	1.82E-21	0	5.50E-46	0	6.46E-72
50	8.46E-28	2.72E-21	0	9.53E-46	0	1.24E-71
55	2.84E-27	4.17E-21	0	1.69E-45	0	2.51E-71
60	7.94E-27	6.46E-21	0	3.06E-45	0	4.70E-71
65	2.14E-26	1.03E-20	0	5.51E-45	0	9.68E-71
70	5.21E-26	1.63E-20	0	9.84E-45	0	2.00E-70
75	1.26E-25	2.66E-20	0	1.98E-44	0	4.60E-70
80	3.13E-25	4.56E-20	0	4.38E-44	0	1.12E-69
85	8.09E-25	8.59E-20	0	1.05E-43	0	3.10E-69
90	2.50E-24	1.99E-19	4.29E-64	3.15E-43	0	1.15E-68
95	9.51E-24	6.63E-19	1.21E-57	1.74E-42	0	8.86E-68
100	6.73E-21	1.32E-15	1.00E-50	3.55E-38	1.32E-84	5.14E-63

and δ = noise probability otherwise. Let p denote the bit insertion probability ($1 -$ clocking probability). The partial probability satisfies the recursion $P(s,e) = P(s,e-1)p + P(s-1,e)(1-p)\delta(a(s), b(e+s))$ for $1 \leq s \leq m$ and $0 \leq e \leq n-m$, with initial values $P(1,e) = p^e, 0 \leq e \leq n-m$, and $P(s,-1) = 0, 2 \leq s \leq m$.

In order to validate the suitability of joint probability as an effective correlation measure for reduced Alpha1, experiments were conducted using a computer. The experiments compared the joint probability values for random strings of lengths n and m, and the values for correlated strings of lengths n and m. The value of m is determined empirically to be $n * p - 2 * \sqrt{n}$. The experiments were conducted for the three string lengths $n = 100, 200$ and 300 bits, respectively. The bit insertion probability is 7/16 and the probability of noise is at 0.75. For each experiment, 10,000 trials were conducted. The results are given in Table 3.

For $n = 100$, the lowest joint probability value for the correlated strings is 1.24E-26. This value is higher than at least 60% of the random strings. For $n = 200$, this increased to at least 95%. Finally, for $n = 300$, the joint probability clearly distinguished the correlated strings and random strings. The results indicated that joint probability is an effective correlation measure.

Attacking *R*4. Let $\{R4(t)\}_{t=0}^{n-1}$ be the binary string produced under some unknown clock control. Assume the clocking probability of $R4$ is 15/28, based on the results presented in [11]. That is,$R4(t)$ has 13/28 probability of being repeated. Assume the distributions of $R2(t)$ and $R3(t)$ are uniformly random, then $R2(t)$ OR $R3(t) = 1$ with probability 0.75. In a correlation attack on $R4$, the input from $R2$ and $R3$ can be considered noise.

The correlation attack on $R4$ requires exhaustive search through all possible initial states of $R4$. For each initial state, a segment of the LFSR sequence of length of length m is produced under regular clocking. Given the known keystream segment $\{z'(t)\}$, a joint probability value can then be calculated for every state. The joint probability for the actual $R4$ initial state should be greater than the joint probability for the vast majority of the incorrect states. Also, from Table 3, we expect the difference between the joint probability value for the correct state and joint probabilities for the incorrect states to increase if the length of known keystream increases. The attack on $R4$ requires exhaustive search over the 2^{35} initial states of $R4$. For each of the 2^{35} $\hat{R}4$ candidate initial states:
Input: reduced Alpha1 keystream $\{z'(t)\}_{t=0}^{n-1}$
Output: the joint probability value of $\hat{R}4$

1. Generate $\{\hat{R}4(t)\}_{t=0}^{n-1}$, the regularly clocked $\hat{R}4$ output sequence.
2. Calculate the joint probability of $\{\hat{R}4(t)\}_{t=0}^{n-1}$ and $\{z'(t)\}_{t=0}^{n-1}$, using the iterative procedure discussed in Section **??**.

The joint probability values are stored and indexed by the candidate initial states. This array is then sorted on the joint probability value. One of the candidate $\hat{R}4$ initial states with the highest joint probability is likely to be the correct initial state.

The joint probability algorithm compares the two input strings, the random binary string and the keystream. This attack therefore has complexity $2^{35} \cdot n^2$. There is a tradeoff between the length of known keystream n and the accuracy of the attack.

In our computer simulations, we calculated the joint probability value for the correct $R4$ initial state and 100,000 other random $R4$ initial states using 7000 bits of keystream. The experiment was repeated 100 times. It is observed that joint probability values associated with the correct $R4$ initial states are always in the top 1% of the values. Using this attack to search through all 2^{35} initial states of $R4$ gave a pool of 2^{28} candidate states. Using a further 7000 bits of keystream to search through the pool reduced the size of the pool down to 2^{21}. For every 7000 bits of keystream used in the attack the pool of candidate initial states is reduced by 2^7. Repeating the attack five times reduced the list to only one candidate. The text requirement of the whole attack is 35,000 bits. The complexity of this attack is $2^{35} \cdot 7000^2 + 2^{28} \cdot 7000^2 + \ldots + 2^7 \cdot 7000^2 \approx 2^{61}$.

4.4 Recovery of $R2$ and $R3$

Assuming the correct initial states of $R1$ and $R4$ have been recovered, the next target in our divide and conquer attack is $R2$. It is targeted because of its shorter length. We exhaustively search through all possible initial states of $R2$ and try to reconstruct $R3$ for each of the $R2$ states. If the wrong $R2$ state is used in the reconstruction of $R3$, the reconstruction will eventually fail. For the correct $R2$ state, successful reconstruction of $R3$ will be possible. The following algorithm is used:
Inputs: reduced Alpha1 keystream $\{z'(t)\}_{t=0}^{n-1}$, initial register state $R4$
Outputs: initial register states $R2$ and $R3$

```
1. Guess $\hat{R2}$, a candidate initial state for $R2$. Set $t = 0$.
2. Calculate $\hat{R3}(t)$:
   – if $z'(t) \oplus R4(t) = 0$ and $\hat{R2}(t) = 0$ then $\hat{R3}(t) = 0$,
   – if $z'(t) \oplus R4(t) = 0$ and $\hat{R2}(t) = 1$ then $\hat{R2}$ is wrong; go to 1,
   – if $z'(t) \oplus R4(t) = 1$ and $\hat{R2}(t) = 0$ then $\hat{R3}(t) = 1$,
   – otherwise guess $\hat{R3}(t)$.
3. Guess the value of two clocking taps, $\hat{R3}_{10}(t)$ and $\hat{R3}_{22}(t)$.
4. Clock $\hat{R2}$, $\hat{R3}$ and $R4$ according to majority function
   – if $\hat{R3}$ did not clock but $\hat{R3}(t) \neq \hat{R3}(t-1)$ then $\hat{R3}$ is wrong; decrement $t$
     and backtrack to 2.
5. If $\hat{R3}$ has clocked fewer than 11 times, increment $t$ and go to 2.
6. Reset $R4$ to initial state. Set $\hat{R2}$ and $\hat{R3}$ to guessed states
   – clock cipher to produce $\hat{z'}(t)$. If $\hat{z'}(t) \neq z'(t)$, $\hat{R2}$ is wrong; go to 1,
   – iterate clocking until $t = n$,
   – output $\hat{R2}$ and $\hat{R3}$.
```

In the algorithm above, all possible clocking taps are tested before changing the guess for the $\hat{R3}(t)$ value. If all the guesses are exhausted, that is, if the reconstruction of $R3$ fails, then the next $\hat{R2}$ state is tested. This algorithm will give a candidate $\hat{R3}$ initial state for the guessed $\hat{R2}$ state after $R3$ clocked 11 times (11 is the distance between $R3_{33}$ and the first clocking tap, $R3_{22}$). This result can be confirmed by running the reduced Alpha1 generator until $R3$ clocks another 22 times (to cycle through $R3$ completely). If the output sequences agree with the observed $z'(t)$, then the guessed $R2$ state and the reconstructed $R3$ state are the correct initial states. The clocking probability of $R3$ is 15/28, so 33 clocks of $R3$ produce 62 bits of output keystream on average, therefore this attack requires about 62 bits of observed keystream on average.

Figure 4 shows the probability tree for step 2 of the algorithm. Wrong guesses for clocking taps, $R3(t)$ and $R2(t)$ could be detected reliably with probability $\frac{1}{8}$. Assuming the guess for $R2$ is correct, the value of $R3(t)$ can be calculated reliably with probability $\frac{4}{8}$. $R3(t)$ has to be guessed with probability $\frac{3}{8}$. The probability that a register does not clock in a given cycle is $\frac{13}{28}$. Wrong guesses for the clocking taps, $R3(t)$ and $R2$ could be detected with probability $\frac{5}{8} \cdot \frac{13}{28} = \frac{65}{224}$ (including the $\frac{1}{8}$ mentioned above).

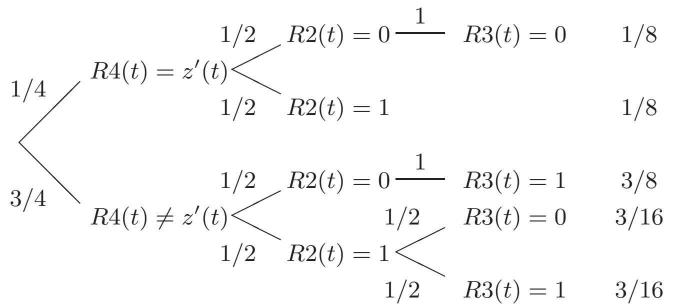


Fig. 4. Probability Tree of Guessing $R3(t)$

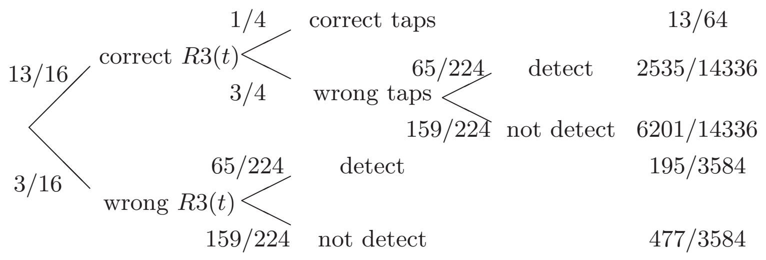


Fig. 5. Probability Tree of Guessing $R3$ Clocking Taps

Figure 5 shows the probability for guessing the correct $R3$, clocking taps and also the probability of detecting wrong guesses. The probability of wrong guesses not being detected is $\frac{8109}{14336} = 0.5656$ at each branch, this means that when the search tree is traversed to the depth of 8, the probability of not detecting wrong branches diminishes to 0.01. The probability of following a correct branch is therefore 0.99. There are 7 wrong branches at each step, giving a search space of $7^8 \approx 2^{22.5}$ in the worst case scenario. Therefore, to break $R2$ and $R3$ in the method described requires $2^{31} \cdot 2^{22.5} = 2^{53.5}$. This is less effort than exhaustive search of the $R2$ and $R3$ initial states (2^{64}), and also less than the complexity of recovering $R4$ (2^{61}).

5 Conclusion

Alpha1 is intended to have an effective key space of 128 bits. However, in this paper we have presented a complete divide and conquer attack which shows the security level is reduced to approximately 61 bits. This attack recovers the initial states of all registers, unlike previous partial attacks which dealt only with the shortest and regularly clocked register.

The attack described in this paper begins with an improved recovery of $R1$, followed by a new probabilistic correlation attack on $R4$, and finally a guess and check attack to recover the initial states of $R2$ and $R3$. The attack on $R1$ involves exhaustive search over the 2^{29} initial states and required around 2048 bits of keystream. The exhaustive attack on $R4$ requires 35,000 bits of keystream. The attack on $R2$ and $R3$ requires exhaustive search over 2^{31} initial states of $R2$ and 62 bits of keystream on average. The text requirement for recovering the secret key of Alpha1 is therefore 35,000 bits. The attacks on $R1$ and $R4$ have complexity 2^{39} and $2^{60.5}$, respectively. The initial states of $R2$ and $R3$ can be recovered in about $2^{53.5}$ operations. This shows that Alpha1 can be broken with complexity 2^{61}, much less than exhaustive key search over a key space of 2^{128}. The lack of security makes Alpha1 an inappropriate candidate for securing mobile and wireless applications.

References

[1] M. Briceno, I. Goldberg, and D. Wagner. A Pedagogical Implementation of A5/1, May 1999. http://www.scard.org. 418

[2] D. Coppersmith, H. Krawczyk, and Y. Mansour. The Shrinking Generator. In D. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993. Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 22–39, Berlin, 1994. Springer-Verlag. 425

[3] J. Golić and R. Menicocci. Edit Distance Correlation Attack on the Alternating Step Generator. In B. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1993. Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 499–512, Berlin, 1997. Springer-Verlag. 419, 425

[4] J. Golić and R. Menicocci. Computation of Edit Probabilities and Edit Distances for the A5-Type Keystream Generator. *Journal of Complexity*, 18(1):356–374, 2002. 419, 425

[5] J. Golić and L. O'Connor. Embedding and Probabilistic Correlation Attacks on Clock-Controlled Shift Registers. In A. DeSantis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994. Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 230–243, Berlin, 1995. Springer-Verlag. 425

[6] N. Komninos, B. Honary, and M. Darnell. An Efficient Stream Cipher for Mobile and Wireless Devices. In B. Honary, editor, *Cryptography and Coding - 8th IMA International Conference, Cirencester, UK, December 17-19, 2001. Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 294–300, Berlin, 2001. Springer-Verlag. 418

[7] R. Menicocci and J. Golić. Edit Probability Correlation Attack on the Bilateral Stop/Go Generator. In M. Walker, editor, *Cryptography and Coding - 7th IMA International Conference, Cirencester, UK, December 20-22, 1999. Proceedings*, volume 1746 of *Lecture Notes in Computer Science*, pages 201–212, Berlin, 1999. Springer-Verlag. 419, 425

[8] C. Mitchell. Remarks on the Security of the Alpha1 Stream Cipher. Technical Report RHUL-MA-2001-8, Royal Holloway, University of London, December 2001. http://www.ma.rhul.ac.uk/techreports/2001/RHUL-MA-2001-8.pdf. 418, 420, 421

[9] T. Siegenthaler. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Transactions on Computers*, C-34(1):81–85, January 1985. 425

[10] L. Simpson, J. Golić, and E. Dawson. A Probabilistic Correlation Attack on the Shrinking Generator. In C. Boyd and E. Dawson, editors, *Information Security and Privacy - 3rd Australasian Conference, ACISP'98, Brisbane, Australia, July 13-15, 1998. Proceedings*, volume 1438 of *Lecture Notes in Computer Science*, pages 147–158, Berlin, 1998. Springer-Verlag. 425

[11] H. Wu. Cryptanalysis of Stream Cipher Alpha1. In L. Batten and J. Seberry, editors, *Information Security and Privacy - 7th Australasian Conference, ACISP 2002, Melbourne, Australia, July 3-5, 2002. Proceedings*, volume 2384 of *Lecture Notes in Computer Science*, pages 169–175, Berlin, 2002. Springer-Verlag. 418, 419, 421, 422, 423, 427

[12] K. Zeng, C. Yang, and T. Rao. Large Primes in Stream Cipher Cryptography. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology - AUSCRYPT '90, International Conference on Cryptology, Sydney, Australia, January 8-11, 1990. Proceedings*, volume 453 of *Lecture Notes in Computer Science*, pages 194–205, Berlin, 1990. Springer-Verlag. 424

[13] K. Zeng, C. Yang, D. Wey, and T. Rao. Pseudorandom Bit Generators in Stream-Cipher Cryptography. *IEEE Computer*, 24(2):8–17, February 1991. 424

New Block Cipher: ARIA

Daesung Kwon[1], Jaesung Kim[2], Sangwoo Park[1], Soo Hak Sung[3], Yaekwon Sohn[2], Jung Hwan Song[4], Yongjin Yeom[1], E-Joong Yoon[1], Sangjin Lee[5], Jaewon Lee[2], Seongtaek Chee[1], Daewan Han[1], and Jin Hong[1]

[1] National Security Research Institute,
161 Gajeong-dong, Yuseong-gu, Daejeon 305-350, Korea
{ds_kwon,psw,yjyeom,yej,chee,dwh,jinhong}@etri.re.kr

[2] International Science Culture Institute,
P. O. Box 200, Socho-gu 137-602, Korea
{ijkim1,shrek52,bokmin48}@lycos.co.kr

[3] Department of Computing information & mathematics
Paichai University, 426-6 Doma-dong, Seo-gu, Daejeon 302-735 Korea
sungsh@mail.pcu.ac.kr

[4] Department of Mathematics, Hanyang University,
17 Haengdang-dong, Seongdong-gu, Seoul 133-791, Korea
camp123@hanyang.ac.kr

[5] Graduate School of Information Security
Korea University, 1, 5-Ka, Anam-dong, Sungbuk-ku, Seoul 136-701, Korea
sangjin@korea.ac.kr

Abstract. In this paper, we propose a 128-bit block cipher ARIA which is an involution substitution and permutation encryption network(SPN). We use the same S-boxes as Rijndael to eliminate defects which are caused by a totally involution structure. In the diffusion layer of ARIA, a 16×16 binary matrix of the maximum branch number 8 is used to avoid some attacks well applied to the reduced round of Rijndael. ARIA uses only basic operations, S-box substitutions and XOR's together with an involution structure so that it can be efficiently implemented on various platforms.

1 Introduction

The block cipher Rijndael[1] has been selected as AES (Advanced Encryption Standard) by NIST in 2000[2] and also selected as a portfolio of NESSIE(New European Schemes for Signature, Integrity and Encryption) project with Camellia[3], MISTY1[4] and SHACAL-256[5] in 2003. The ciphers, Rijndael and Camellia have 128-bit block size and 128-, 192-, and 256-bit key lengths which are the interface specifications of AES, are considered as representative ciphers which satisfy such the interface. Rijndael[1] is an SPN cipher and uses operations such as multiplications in a finite field to have a diffusion effect of the states in 32-bits. Camellia[3] is a Feistel cipher and uses an 8×8 binary matrix to have a diffusion effect of the states in 64-bits.

In this paper, we propose a 128-bit block cipher ARIA which is an involution SPN which has been studied by AES developers. Khazad[6] and Anubis[7], which has been submitted to NESSIE project[8], are involution SPN ciphers and have involution substitution and diffusion layers by using operations such as multiplications in a finite field. Those variations are favorable in considering the efficiency, but some flaws[9] caused by the totally involution structure have been found. ARIA is an involution SPN block cipher which is not totally involutional. More precisely, the substitution layers are not involution. Therefore, the flaws given in [9] cannot be applied to ARIA. ARIA uses an involutional diffusion layer which is a 16×16 binary matrix. Also ARIA uses two kinds of S-boxes S_1, S_2 and two types of substitution layers $(LS, LS, LS, LS), (LS^{-1}, LS^{-1}, LS^{-1}, LS^{-1})$ where $LS = (S_1, S_2, S_1^{-1}, S_2^{-1})$. The substitution layers are arranged for ARIA to be an involution structure described as follows. In each odd round, the substitution layer is (LS, LS, LS, LS), in each even round, the substitution layer is $(LS^{-1}, LS^{-1}, LS^{-1}, LS^{-1})$. In addition to the involution SPN structure, we choose a diffusion layer to resist against powerful attacks which have been applied to reduced round Rijndael and Camellia, such as collision attacks[10], partial sum attacks[11], and truncated differential attacks. Since the size of the states mixed by the diffusion layers of Rijndael and Camellia are only a quarter or a half the size of a block, attacks which are listed the above are valid for some reduced rounds of those block ciphers. A diffusion layer which mixes all states is one of design principles and it is selected among 16×16 matrices. Note that a 16×16 matrix with entries $a_{ij} \in F$, where F is a finite field that is not $GF(2)$ is not a good choice because they are heavily involved with multiplications in a finite field F. Therefore 16×16 *binary* matrices are only candidates.

The maximum branch number of an invertible 16×16 binary matrix is *8* and the maximum branch number of an invertible 16×16 matrix with entries in a finite field is 17, which will be published later. We construct such an involution binary matrix of branch number *8*, and use some mathematical techniques to find a form in product of matrices, which is for increasing efficiency in 8-bit and 32-bit software implementations.

The cipher ARIA is an involution SPN cipher without having any weakness in S-boxes unlike Anubis and Khazad. The specifications of ARIA in detail are given in Section 2 and motivations of the design are given in Section 3. In Section 4, techniques of efficient implementation for 8-bit/32-bit processors are given. Security analysis against known attacks are given in Section 5.

2 Specifications

2.1 Notations

We use the following notations for describing ARIA.

– $S_i(x)$: The output of S-box $S_i(i = 1, 2)$ for an input x
– $A(x)$: The output of diffusion layer for an input x
– $\oplus$: A bitwise XOR operation

- $\|$: Concatenation of two operands
- $\ggg n$: Right circular rotation of operand by n bits
- $\lll n$: Left circular rotation of operand by n bits
- $\cdot$: Multiplication of two operands, matrix and vector, or two matrices

2.2 The Round Transformation

Each round of the cipher consists of the following three parts.

1. Round key addition: This is done by XORing the 128-bit round key.
2. Substitution layer: There shall be two types of substitution layers.
3. Diffusion layer: A simple linear map which is an involution.

Note that the diffusion layer of the last round is replaced by a round key addition.

Substitution Layer. We use 2 S-boxes S_1, S_2 and their inverses S_1^{-1}, S_2^{-1}. The each S-box is defined to be an affine transformation of the inversion function over GF(2^8).

$$S: \text{ GF}(2^8) \to \text{GF}(2^8),$$
$$S_1\, x \mapsto A \cdot x^{-1} \oplus a,$$

where

$$A = \begin{pmatrix} 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1\\ 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1 \end{pmatrix} \quad \text{and} \quad a = \begin{pmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{pmatrix}.$$

and

$$S_2\, x \mapsto B \cdot x^{247} \oplus b,$$

where

$$B = \begin{pmatrix} 0&1&0&1&1&1&1&0\\ 0&0&1&1&1&1&0&1\\ 1&1&0&1&0&1&1&1\\ 1&0&0&1&1&1&0&1\\ 0&0&1&0&1&1&0&0\\ 1&0&0&0&0&0&0&1\\ 0&1&0&1&1&1&0&1\\ 1&1&0&1&0&0&1&1 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 0\\1\\0\\0\\0\\1\\1\\1 \end{pmatrix}.$$

The precalculated values of S_1, S_2 and S_1^{-1}, S_2^{-1} are given in Table 1 and Table 2. For example, $S_1(\texttt{0x00}) = \texttt{0x63}$, $S_1(\texttt{0x05}) = \texttt{0x6b}$, and $S_1(\texttt{0x72}) = \texttt{0x40}$.

ARIA has two types of S-box layers as shown in Figure 1.

The two types are used alternately and we use an even number of rounds so as to make the cipher involution. Type 1 is used in the odd rounds and type 2 is used in the even rounds.

Table 1. S-box S_1 and S_1^{-1}

S-box S_1

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-box S_1^{-1}

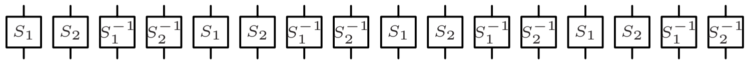
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Diffusion Layer. The diffusion layer $A:\ \mathrm{GF}(2^8)^{16} \to \mathrm{GF}(2^8)^{16}$ is given by

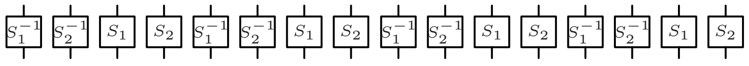
$$(x_0, x_1, \ldots, x_{15}) \mapsto (y_0, y_1, \ldots, y_{15}),$$

where

$$\begin{aligned}
y_0 &= x_3 \oplus x_4 \oplus x_6 \oplus x_8 \oplus x_9 \oplus x_{13} \oplus x_{14}, & y_8 &= x_0 \oplus x_1 \oplus x_4 \oplus x_7 \oplus x_{10} \oplus x_{13} \oplus x_{15},\\
y_1 &= x_2 \oplus x_5 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{12} \oplus x_{15}, & y_9 &= x_0 \oplus x_1 \oplus x_5 \oplus x_6 \oplus x_{11} \oplus x_{12} \oplus x_{14},\\
y_2 &= x_1 \oplus x_4 \oplus x_6 \oplus x_{10} \oplus x_{11} \oplus x_{12} \oplus x_{15}, & y_{10} &= x_2 \oplus x_3 \oplus x_5 \oplus x_6 \oplus x_8 \oplus x_{13} \oplus x_{15},\\
y_3 &= x_0 \oplus x_5 \oplus x_7 \oplus x_{10} \oplus x_{11} \oplus x_{13} \oplus x_{14}, & y_{11} &= x_2 \oplus x_3 \oplus x_4 \oplus x_7 \oplus x_9 \oplus x_{12} \oplus x_{14},\\
y_4 &= x_0 \oplus x_2 \oplus x_5 \oplus x_8 \oplus x_{11} \oplus x_{14} \oplus x_{15}, & y_{12} &= x_1 \oplus x_2 \oplus x_6 \oplus x_7 \oplus x_9 \oplus x_{11} \oplus x_{12},\\
y_5 &= x_1 \oplus x_3 \oplus x_4 \oplus x_9 \oplus x_{10} \oplus x_{14} \oplus x_{15}, & y_{13} &= x_0 \oplus x_3 \oplus x_6 \oplus x_7 \oplus x_8 \oplus x_{10} \oplus x_{13},\\
y_6 &= x_0 \oplus x_2 \oplus x_7 \oplus x_9 \oplus x_{10} \oplus x_{12} \oplus x_{13}, & y_{14} &= x_0 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_9 \oplus x_{11} \oplus x_{14},\\
y_7 &= x_1 \oplus x_3 \oplus x_6 \oplus x_8 \oplus x_{11} \oplus x_{12} \oplus x_{13}, & y_{15} &= x_1 \oplus x_2 \oplus x_4 \oplus x_5 \oplus x_8 \oplus x_{10} \oplus x_{15}.
\end{aligned}$$


S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1}

(a) S-box layer type 1


S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2 S_1^{-1} S_2^{-1} S_1 S_2

(b) S-box layer type 2

Fig. 1. Two types of S-box layers

Table 2. S-box S_2 and S_2^{-1}

S-box S_2

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	e2	4e	54	fc	94	c2	4a	cc	62	0d	6a	46	3c	4d	8b	d1
1	5e	fa	64	cb	b4	97	be	2b	bc	77	2e	03	d3	19	59	c1
2	1d	06	41	6b	55	f0	99	69	ea	9c	18	ae	63	df	e7	bb
3	00	73	66	fb	96	4c	85	e4	3a	09	45	aa	0f	ee	10	eb
4	2d	7f	f4	29	ac	cf	ad	91	8d	78	c8	95	f9	2f	ce	cd
5	08	7a	88	38	5c	83	2a	28	47	db	b8	c7	93	a4	12	53
6	ff	87	0e	31	36	21	58	48	01	8e	37	74	32	ca	e9	b1
7	b7	ab	0c	d7	c4	56	42	26	07	98	60	d9	b6	b9	11	40
8	ec	20	8c	bd	a0	c9	84	4	49	23	f1	4f	50	1f	13	dc
9	d8	c0	9e	57	e3	c3	7b	65	3b	02	8f	3e	e8	25	92	e5
a	15	dd	fd	17	a9	bf	d4	9a	7e	c5	39	67	fe	76	9d	43
b	a7	e1	d0	f5	68	f2	1b	34	70	05	a3	8a	d5	79	86	a8
c	30	c6	51	4b	1e	a6	27	f6	35	d2	6e	24	16	82	5f	da
d	e6	75	a2	ef	2c	b2	1c	9f	5d	6f	80	0a	72	44	9b	6c
e	90	b	5b	33	7d	5a	52	f3	61	a1	f7	b0	d6	3f	7c	6d
f	ed	14	e0	a5	3d	22	b3	f8	89	de	71	1a	af	ba	b5	81

S-box S_2^{-1}

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	30	68	99	1b	87	b9	21	78	50	39	db	e1	72	9	62	3c
1	3e	7e	5e	8e	f1	a0	cc	a3	2a	1d	fb	b6	d6	20	c4	8d
2	81	65	f5	89	cb	9d	77	c6	57	43	56	17	d4	40	1a	4d
3	c0	63	6c	e3	b7	c8	64	6a	53	aa	38	98	0c	f4	9b	ed
4	7f	22	76	af	dd	3a	0b	58	67	88	06	c3	35	0d	01	8b
5	8c	c2	e6	5f	02	24	75	93	66	1e	e5	e2	54	d8	10	ce
6	7a	e8	8	2c	12	97	32	ab	b4	27	0a	23	df	ef	ca	d9
7	b8	fa	dc	31	6b	d1	ad	19	49	bd	51	96	ee	e4	a8	41
8	da	ff	cd	55	86	36	be	61	52	f8	bb	0e	82	48	69	9a
9	e0	47	9e	5c	04	4b	34	15	79	26	a7	de	29	ae	92	d7
a	84	e9	d2	ba	5d	f3	c5	b0	bf	a4	3b	71	44	46	2b	fc
b	eb	6f	d5	f6	14	fe	7c	70	5a	7d	fd	2f	18	83	16	a5
c	91	1f	05	95	74	a9	c1	5b	4a	85	6d	13	07	4f	4e	45
d	b2	0f	c9	1c	a6	bc	ec	73	90	7b	cf	59	8f	a1	f9	2d
e	f2	b1	00	94	37	9f	d0	2e	9c	6e	28	3f	80	f0	3d	d3
f	25	8a	b5	e7	42	b3	c7	ea	f7	4c	11	33	03	a2	ac	60

An equivalent expression would be given by a matrix multiplication as follow.

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \\ y_{13} \\ y_{14} \\ y_{15} \end{pmatrix} = \begin{pmatrix} 0&0&0&1&1&0&1&0&1&1&0&0&0&1&1&0 \\ 0&0&1&0&0&1&0&1&1&1&0&0&1&0&0&1 \\ 0&1&0&0&1&0&1&0&0&0&1&1&1&0&0&1 \\ 1&0&0&0&0&1&0&1&0&0&1&1&0&1&1&0 \\ 1&0&1&0&0&1&0&0&1&0&0&1&0&0&1&1 \\ 0&1&0&1&1&0&0&0&0&1&1&0&0&0&1&1 \\ 1&0&1&0&0&0&0&1&0&1&1&0&1&1&0&0 \\ 0&1&0&1&0&0&1&0&1&0&0&1&1&1&0&0 \\ 1&1&0&0&1&0&0&1&0&0&1&0&0&1&0&1 \\ 1&1&0&0&0&1&1&0&0&0&0&1&1&0&1&0 \\ 0&0&1&1&0&1&1&0&1&0&0&0&0&1&0&1 \\ 0&0&1&1&1&0&0&1&0&1&0&0&1&0&1&0 \\ 0&1&1&0&0&0&1&1&0&1&0&1&1&0&0&0 \\ 1&0&0&1&0&0&1&1&1&0&1&0&0&1&0&0 \\ 1&0&0&1&1&1&0&0&0&1&0&1&0&0&1&0 \\ 0&1&1&0&1&1&0&0&1&0&1&0&0&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \end{pmatrix} \tag{1}$$

2.3 Key Schedule

The key schedule of ARIA consists of two parts, which are initialization and round key generation as follows.

Initialization. In the initialization part, four 128-bit values W_0, W_1, W_2, W_3 are generated from the master key MK, by using a 3-round 256-bit Feistel cipher.

Note that MK can be of 128, 192, or 256 bit size. We first fill out the 128-bit value KL with bits from MK and use what is left of MK (if any) on the 128-bit value KR. The space remaining on KR (if any) is filled with zero as the follow.

$$KL||KR = MK||0 \cdots 0.$$

Then we set

$$W_0 = KL, \qquad W_2 = F_e(W_1, CK_2) \oplus W_0,$$
$$W_1 = F_o(W_0, CK_1) \oplus KR, \qquad W_3 = F_o(W_2, CK_3) \oplus W_1.$$

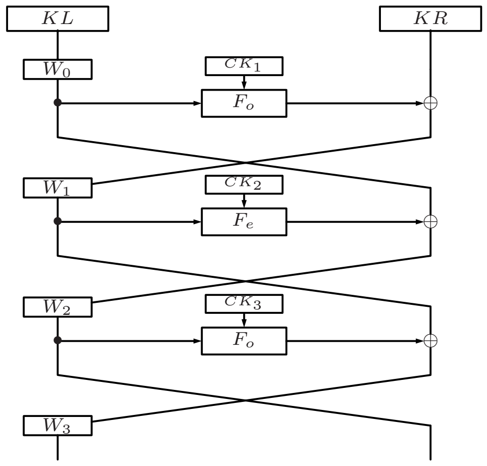


Fig. 2. Initialization part of key schedule

Here, F_o and F_e are even and odd round functions, respectively, given in the previous section. The 128-bit keys CK_i of the round functions are fixed to be the rational part of π^{-1} and are given as follows.

$$CK_1 = \texttt{0x517cc1b727220a94fe13abe8fa9a6ee0}$$
$$CK_2 = \texttt{0x6db14acc9e21c820ff28b1d5ef5de2b0}$$
$$CK_3 = \texttt{0xdb92371d2126e9700324977504e8c90e}$$

This initialization process is given in Figure 2.

Round Key Generation. In the generation part, combining the four values W_i to obtain an encryption round key ek_i and the decryption round key dk_i. Note that the number of rounds we use are 10, 12, or 14 which are depending on the size 128, 192, or 256 of the master key. Since there is one more key addition layer in the last round, 128-bit round keys are needed in the $11^{th}, 13^{th}$, or 15^{th} round, depending on the size of master key. The value KR is also used in generating the round keys when the master key is of 256-bit size.

$$\begin{aligned}
ek_1 &= (W_0^{\ggg 7}) \oplus (W_1^{\lll 11}), & ek_2 &= (W_1^{\lll 22}) \oplus (W_2),\\
ek_3 &= (W_2^{\ggg 17}) \oplus (W_3^{\lll 16}), & ek_4 &= (W_0^{\ggg 14}) \oplus (W_3^{\lll 32}),\\
ek_5 &= (W_0^{\ggg 21}) \oplus (W_2^{\ggg 34}), & ek_6 &= (W_1^{\lll 33}) \oplus (W_3^{\lll 48})\\
ek_7 &= (W_1^{\lll 44}) \oplus (W_2^{\ggg 51}), & ek_8 &= (W_0^{\ggg 28}) \oplus (W_3^{\lll 64}),\\
ek_9 &= (W_1^{\lll 55}) \oplus (W_3^{\lll 80}), & ek_{10} &= (W_0^{\ggg 35}) \oplus (W_2^{\ggg 68}),\\
ek_{11} &= (W_0^{\ggg 42}) \oplus (W_1^{\lll 66}), & ek_{12} &= (W_1^{\lll 77}) \oplus (W_2^{\ggg 85}) \oplus (W_3^{\lll 96}),\\
ek_{13} &= (W_0^{\ggg 49}) \oplus (W_2^{\ggg 102}), & ek_{14} &= (W_2^{\ggg 119}) \oplus (W_3^{\lll 112}) \oplus (KR^{\lll 64}),\\
ek_{15} &= (W_0^{\ggg 56}) \oplus (W_1^{\lll 88}) \oplus (KR).
\end{aligned}$$

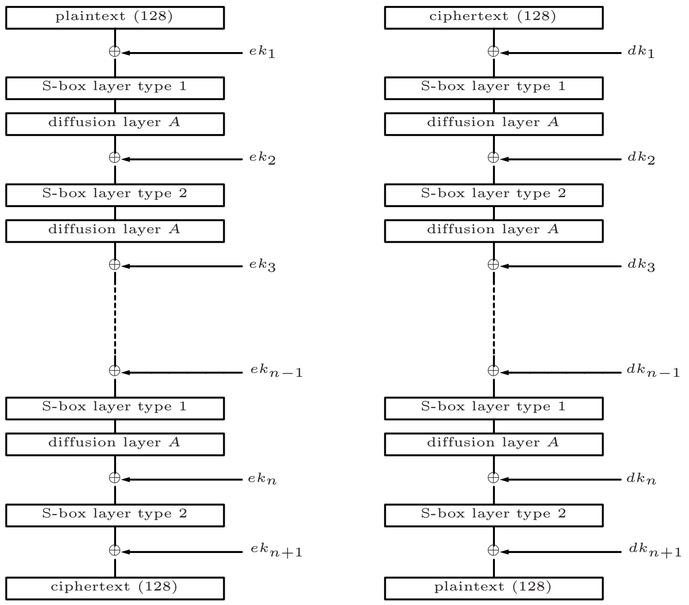


Fig. 3. Encryption and decryption processes

The decryption round keys are different from the encryption round keys and are derived from the encryption round keys. The ordering of round keys are reversed followed by the output of the diffusion layer A to all round keys except for the first and the last. The following equations represent how the decryption keys are computed by n which is given as the number of rounds.

$$dk_1 = ek_{n+1}, dk_2 = A(ek_n), dk_3 = A(ek_{n-1}), \cdots, dk_n = A(ek_2), dk_{n+1} = ek_1.$$

2.4 The Cipher

The encryption and decryption processes of an n-round ARIA, where n is even, are given in Figure 3.

Note that the above two processes are identical except in the use of round keys.

2.5 Number of Rounds

The number of rounds n can be a multiple of 2. We recommend the number of rounds of ARIA by 10-/12-/14-rounds for 128-/192-/256-bit keys, respectively.

3 Motivation for Design Choices

A cryptographic algorithm is usually designed so as to be implemented efficiently in both software and hardware. However, there is no set of rules one can follow to achieve this goal. We have designed ARIA taking into account many elements such as gate counts, memory requirements in smart card implementations, and software performance on multiple platforms.

The cipher consists only of four 8×8 substitution tables (S-boxes) and a linear transformation which can be efficiently implemented even in 8-bit low-end smart cards. The linear transformation, used as the diffusion layer, is an involution 16×16 binary matrix. It has the maximum branch number 8, and the number of ones has been minimized so as to reduce the total number of logical operations. We use basic logical operations in order to ease of hardware implementations.

3.1 S-Box

We design the S-boxes to meet the following criteria.

- The size of input and output is 8. i.e., it should be a function over GF(2^8).
- Degree of the Boolean polynomial describing each output bit is 7.
- It has the smallest of the maximum probability 2^{-6} for differential characteristic and linear approximation.

The S-box S on the inversion function over GF(2^8) is used to satisfy the above criteria. High degree of the Boolean polynomial representing each output bit of the S-boxes makes difficult to be applied higher order differential attacks on the cipher.

An affine transformation which is an 8×8 binary matrix followed by the inversion function is used for getting rid of fixed points and is chosen with the maximum branch number 5.

3.2 Diffusion Layer

In designing a cipher, choosing a diffusion layer is important in both efficiency and security aspects. Since we want the cipher to be involutional and all states to be mixed by the diffusion layer, we search a matrix in 16×16 binary matrices. We choose a matrix which satisfies the following properties.

- It should be involutional.
- The branch number should be maximal.
- It should be suitable for implementations on 8-bit/32-bit processors.
- It should be suitable for hardware implementations.

The branch number $\beta(A)$ of a $n \times n$ diffusion layer A over a finite field F is defined by

$$\beta(A) = \min\{wt(x) + wt(Ax^T) | x \in F^n, x \neq 0\},$$

where $wt(x)$ is the Hamming weight of x.

We proved that the maximum branch number of invertible 16×16 binary matrices is 8 while the maximum branch number of invertible 16×16 matrices over finite field $GF(2^8)$ is 17. This will be published later. Since the matrix should be involutional, we searched the matrices of the form

$$M_1^{-1} \cdot M_2 \cdot M_1$$

where M_2 is an involution block diagonal matrix whose diagonal consists of four 4×4 binary matrices and M_1 is any 16×16 binary matrix which can be written as a 4×4 binary matrix on the vectors with four 32-bits block entries. The restriction are set for the efficiency on 32-bit processors.

For many matrices of the above form, we examined whether the branch number is 8 and the Hamming weights of each column is 7. More description of the matrix will be given in Section 4 and in [12].

3.3 Key Expansion

The key schedule is designed by the following criteria.

- Input to the key schedule is of 128, 192, or 256 bit size.
- Only the basic logical operations are used.
- Resistance to related key attacks.
- It should be difficult to recover the master key from a small fraction of round keys.
- Delay time to the encryption or decryption process caused by key schedule should be minimized.

We use a 256-bit Feistel type cipher whose core function is the same as our round function to obtain four 128-bit values from master key. Then each round key is derived from two of the four values and the master key by rotating and XORing the 128-bit values. The size of each rotation was designed so as to deter one from obtaining one round key from a combination of other round keys.

In the key expansion part, the values W_0, W_1, W_2, W_3 and the rotation bits were chosen to satisfy the following conditions.

- In any two consecutive rounds, at least one of the 128-bit values used to generate the round keys is not common.
- For any two round keys that were generated from the same set of 128-bit values, one should not be able to recover a part of one round key from a part of the other round key.
- Calculating the first round key should cause as little as possible delay time to the encryption or decryption process.

4 Implementation Aspects

ARIA is suitable for efficient implementations on various environments, especially in hardware. We present some techniques[12] which optimize implementations on 8-bit processors such as Smart Cards and on 32-bit processors such as PCs.

4.1 8-bit Based Implementations

On an 8-bit processor, except for S-box substitution, all operations are XOR's. The implementation of S-box substitution requires four tables of 256 bytes.

In a straight coding, 112 XOR's are required to implement one round except for S-box substitutions. We present a method to reduce the number of operations to 76 XOR's using four additional variables $T_1, \cdots, T_4$ as follows.

$$\begin{array}{ll}
T_1 = x_4 \oplus x_5 \oplus x_{10} \oplus x_{15}, & T_2 = x_3 \oplus x_6 \oplus x_9 \oplus x_{16}, \\
y_1 = x_7 \oplus x_9 \oplus x_{14} \oplus T_1, & y_2 = x_8 \oplus x_{10} \oplus x_{13} \oplus T_2, \\
y_6 = x_2 \oplus x_{11} \oplus x_{16} \oplus T_1, & y_5 = x_1 \oplus x_{12} \oplus x_{15} \oplus T_2, \\
y_{12} = x_3 \oplus x_8 \oplus x_{13} \oplus T_1, & y_{11} = x_4 \oplus x_7 \oplus x_{14} \oplus T_2, \\
y_{15} = x_1 \oplus x_6 \oplus x_{12} \oplus T_1, & y_{16} = x_2 \oplus x_5 \oplus x_{11} \oplus T_2,
\end{array}$$

$$\begin{array}{ll}
T_3 = x_2 \oplus x_7 \oplus x_{12} \oplus x_{13}, & T_4 = x_1 \oplus x_8 \oplus x_{11} \oplus x_{14}, \\
y_3 = x_5 \oplus x_{11} \oplus x_{16} \oplus T_3, & y_4 = x_6 \oplus x_{12} \oplus x_{15} \oplus T_4, \\
y_8 = x_4 \oplus x_9 \oplus x_{14} \oplus T_3, & y_7 = x_3 \oplus x_{10} \oplus x_{13} \oplus T_4, \\
y_{10} = x_1 \oplus x_6 \oplus x_{15} \oplus T_3, & y_9 = x_2 \oplus x_5 \oplus x_{16} \oplus T_4, \\
y_{13} = x_3 \oplus x_8 \oplus x_{10} \oplus T_3, & y_{14} = x_4 \oplus x_{70} \oplus x_9 \oplus T_4.
\end{array}$$

If it is implemented serially, only one extra variable is required in the method.

4.2 Software Implementations on 32-bit processors

On 32-bit processors, Rijndael and Camellia can be implemented efficiently by combining S-box substitutions and the diffusion layer by 8×32 table lookups. This technique is suitable for a diffusion layer which is based on 32-bits. Since the diffusion layer of ARIA is based on 128-bits, on 32-bit processors, it looks less efficient than Rijndael and Camellia. However, we choose a matrix in 16×16 binary matrices which can be efficiently implemented on 32-bit processors.

Since M_1 is an involution($M_1^{-1} = M_1$ in Section 3.2) The diffusion layer A is chosen in the form of $M_1 \cdot M_2 \cdot M_1$ to have an involution structure where

$$M_1 = \begin{pmatrix} I & I & I & 0 \\ I & 0 & I & I \\ I & I & 0 & I \\ 0 & I & I & I \end{pmatrix}, \quad M_2 = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 \\ 0 & 0 & 0 & P_3 \end{pmatrix} \cdot \begin{pmatrix} T & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{pmatrix},$$

for the 4×4 identity matrix I and following four 4×4 matrices

$$T = \begin{pmatrix} 0&1&1&1 \\ 1&0&1&1 \\ 1&1&0&1 \\ 1&1&1&0 \end{pmatrix}, \quad P_1 = \begin{pmatrix} 0&1&0&0 \\ 1&0&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \end{pmatrix}, \quad P_2 = \begin{pmatrix} 0&0&1&0 \\ 0&0&0&1 \\ 1&0&0&0 \\ 0&1&0&0 \end{pmatrix}, \quad P_3 = \begin{pmatrix} 0&0&0&1 \\ 0&0&1&0 \\ 0&1&0&0 \\ 1&0&0&0 \end{pmatrix}.$$

For simplifying the above notations, we use the following notations

$$P = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & P_1 & 0 & 0 \\ 0 & 0 & P_2 & 0 \\ 0 & 0 & 0 & P_3 \end{pmatrix}, \quad M = \begin{pmatrix} T & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{pmatrix}.$$

If we let S be the S-box substitution, the one round except for key addition can be written as $A \cdot S = M_1 \cdot M_2 \cdot M_1 \cdot S$. It is easy to see that the form of $M_1 \cdot S$ is not implemented efficiently by 8×32 table lookups. So, we make the following modification for the representation of the diffusion layer as follows:

$$\begin{aligned} A &= M_1 \cdot M_2 \cdot M_1 = M_1 \cdot P \cdot M \cdot M_1 = M_1 \cdot P \cdot M \cdot M_1 \cdot M \cdot M \\ &= M_1 \cdot P \cdot M \cdot M \cdot M_1 \cdot M = M_1 \cdot P \cdot M_1 \cdot M. \end{aligned}$$

Since M is a block diagonal matrix, $M \cdot S$ can be implemented by 8×32 table lookups. It is clear that M_1 is implemented by simple 32-bit word operations. The operation P is a byte-oriented operation that is done within each 32-bit word. Hence this gives a way to implement ARIA on a 32-bit based machine. On 32-bit processors, the encryption speed of ARIA is at least 70% of that of Rijndael. Notice that there are only three or four ciphers submitted in AES and NESSIE project whose encryption speeds are in the above range. Therefore we can see that ARIA is also efficient on 32-bit processors.

4.3 Remark on The Hardware Implementations

The hardware length is mainly determined by the number of S-box layers unless diffusion layers are too heavy. Since the number of S-box layers are same as that of Rijndael and smaller than that of Camellia, throughput is almost same as that of Rijndael and faster than that of Camellia. Since ARIA is involutional, it requires only one procedure to be implemented for the encryption and the decryption unlike Rijndael. Therefore the area to be implemented in hardware for ARIA is smaller than that of Rijndael, i.e., in considering the efficiency, ARIA is better than Rijndael.

5 Strength Against Known Attacks

5.1 Differential and Linear Cryptanalysis

The maximum differential probability p and the maximum linear probability q of the S-box S for ARIA is given by

$$p = q = 2^{-6}.$$

The branch number β_d in respect to the differential cryptanalysis and the branch number β_l in respect to the linear cryptanalysis for the diffusion layer A of ARIA are given by

$$\beta_d = \beta_l = 8.$$

The minimum number of active S-boxes with respect to differential and linear cryptanalysis in r-rounds is

$$8 \cdot \lfloor r/2 \rfloor + 2 \cdot (r/2 - \lfloor r/2 \rfloor).$$

The upper bound on differential characteristic probabilities and linear approximation probabilities for 6-rounds is $(2^{-6})^{24} = 2^{-144}$. Therefore, there are no effective differential characteristics and linear approximations for ARIA in 6 or more rounds.

The upper bound on differential characteristic probabilities and linear approximation probabilities for 5-rounds is $(2^{-6})^{17} = 2^{-102}$ which is greater than 2^{-128}. Even if an attacker uses the 5-round differential characteristic or linear approximation on ARIA we expect 8 or more rounds to provide sufficient resistance against differential and linear cryptanalysis.

5.2 Truncated Differential Cryptanalysis

By computational experiments, we searched effective truncated differential characteristics which distinguish the reduced-rounds of ARIA from a random permutation. We found many effective truncated differential characteristics on 5 round variant of ARIA which can be used to attack 7 rounds. However, there are no effective truncated differential characteristics on 6 or more rounds. Thus, we think that ARIA has sufficient security against truncated differential cryptanalysis.

5.3 Impossible Differential Cryptanalysis

We have confirmed that there are no bytes whose difference is always zero(or nonzero) after 2-rounds of encryption. Also, We have checked that there are no bytes whose difference is always zero or nonzero after 2-rounds of decryption. Therefore, we expect that there are no impossible differentials on 4 or more rounds.

5.4 Square Attack (Integral Cryptanalysis)

Applying the square attack on ARIA, we considered a collection of plaintexts in which each byte is either active or passive.

In case of ARIA, for any collection of plaintexts, determining whether any given byte position is balanced or not after 3-rounds of encryption is not possible. Therefore, one can construct only 2-round distinguishers.

5.5 Higher Order Differential Cryptanalysis

The S-boxes S and S^{-1} of ARIA have algebraic degree 7. Each output bit of the S-box can be regarded as a Boolean function with 8 input variables. After three rounds the algebraic degree of any intermediate bit becomes 7^3. Thus, the number of plaintexts needed for higher order differential attack using a 3 round distinguishers is greater than 2^{128}. So only up to 2 round distinguisher may be found for application of higher order differential attack.

5.6 Interpolation Attack

In the interpolation attack, using plaintext and ciphertext pairs, the attacker constructs polynomials describing relations between the input and output to the cipher. If the constructed polynomials have small degree, only a small number of plaintexts and their corresponding ciphertexts are necessary to solve for the coefficients of the polynomial. This is done through the use of Lagrange interpolation formula[14].

We expect that the complicated expression of the S-boxes used in ARIA, together with the effect of the diffusion layer will prohibit the interpolation attack on more than just a few rounds.

5.7 Weakness in The Key Schedule

Since the round keys in ARIA are always applied using the XOR operation, one cannot find a weak key class as in IDEA. Also, we expect that there are no equivalent keys. Thus there is no restriction on key selection. The key schedule of ARIA has sufficient number of nonlinear components so as to avoid the related key attack or any other attacks rising from the weakness of the key schedule. In particular, it is impossible to calculate the encryption key from partial information of round keys.

6 Conclusion

We have proposed a new 128-bit block cipher ARIA based on SPN structure. ARIA is a simple and elegant algorithm with the following properties:

- uses only basic operations such as XOR and S-box substitution.
- uses a 16×16 binary matrix with branch number 8(maximal) in diffusion layer.
- adopts an involutional structure for efficient implementations on multiple environments.
- does not adopt a totally involutional structure to avoid flaws found in Khazad and Anubis.

We could not find any significant weakness and have not inserted any hidden weakness. We think that ARIA is suitable for most platforms and can be widely used.

References

[1] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer, 2001. 432
[2] NIST, NIST announces that Rijndael has been selected as the proposed AES. October 2, 2000. Available at http://csrc.nist.gov/CryptoToolkit/aes/ 432

[3] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Camellia: A 128-bit block cipher suitable for multiple platforms - design and analysis, LNCS 2012 , pages 39–56. Springer, 2000. 432

[4] M. Matsui, *Block Encryption Algorithm MISTY*, *Fast Software Encryption*, 4th InternationalWorkshop Proceeding, LNCS 1267, Springer-Verlag, pp.54-68, 1997. 432

[5] H.Handschuh, D.Naccache, SHACAL, In proceedings of the First Open NESSIE Workshop, November 2000. 432

[6] P. S. L. M. Barreto and V. Rijmen, *T*he Khazad legacy-level block cipher. Primitive submitted to NESSIE, Sept. 2000. 433

[7] P. S. L. M. Barreto and V. Rijmen, *T*he Anubis block cipher. Primitive sub- mitted to NESSIE, Sept. 2000. 433

[8] NESSIE Project, New European Schemes for Signatures, Integrity and Encryption, Homepage-avaiable at http://cryptonessie.org. 433

[9] A. Biryukov, *Analysis of Involutional Ciphers: Khazad and Anubis*, In Fast Software Encryption, Proceedings of FSE 2003. 433

[10] H. Gilbert and M. Minier, *A collision attack on seven rounds of Rijndael*, Proceeding of the third AES conference, pp230–241, NIST, 2000. 433

[11] N. Ferguson, J. Kesley, S. Lucks, B. Schneier, M. Stay, D. Wagner and F. Whiting, *Improved Cryptanalysis of Rijndael*, In Fast Software Encryption, Proceedings FSE 2000, LNCS #1978, pp. 213–230, Springer-Verlag, 2000. 433

[12] BonWook Koo, HwanSeok Jang, JungHwan Song. Constructing and Cryptanalysis of a 16x16 Binary Matrix as a Diffusion Layer. , editors, *WISA2003*, *Lecture Notes in Computer Science*, Springer, 2003. 440

[13] David Wagner. The boomerang attack. In Lars R. Knudsen, editor, *Preproceedings of Fast Software Encryption Workshop 1999*, pages 155–169, 1999.

[14] Thomas Jakobsen and Lars R. Knudsen. The interpolation attack on block ciphers. In Eli Biham, editor, *Preproceedings of Fast Software Encryption Workshop 1997*, pages 28–40, 1997. 444

Truncated Differential Attacks on 8-Round CRYPTON

Jongsung Kim[1], Seokhie Hong[1], Sangjin Lee[1],
Junghwan Song[2], and Hyungjin Yang[1]

[1] Center for Information Security Technologies(CIST)
Korea University, Anam Dong, Sungbuk Gu, Seoul, Korea
{joshep,hsh,sangjin}@cist.korea.ac.kr, yangh@korea.ac.kr
[2] Department of Mathematics, Hanyang University,
17 Haengdangdong Seongdongku, Seoul, Korea
camp123@hanyang.ac.kr

Abstract. CRYPTON [8] is a 12-round block cipher proposed as an AES candidate by C.H. Lim in 1998. C.H. Lim also introduced in 1999 [9] a modified CRYPTON with a new key schedule and new S-boxes. This paper presents truncated differential attacks on the modified CRYPTON up to 8 out of its 12 rounds. These are the best known attacks for this cipher.

Keywords. Block Cipher, Truncated Differential Attack, CRYPTON

1 Introduction

The block cipher CRYPTON, designed based on the structure of SQUARE [4], was proposed as a candidate algorithm for the Advanced Encryption Standard (AES). However, to ensure a higher level of security and to fix some minor weakness in the key schedule, the algorithm designer made some changes in the S-box construction and key scheduling for the AES proposal. This paper is dedicated to the analysis of the modified version of CRYPTON. Throughout this paper, we will just call this modified version CRYPTON.

The main cryptanalytic results obtained on CRYPTON so far are the analysis of the square attack on 6-round CRYPTON [6] and the impossible differential attacks on 5-round [13] and 6-round CRYPTON [3]. (These square and impossible differential attacks can be applied to both the initial version and the modified version of CRYPTON.)

The linear part of CRYPTON consists of a bit permutation π and a bytes transposition τ. From [1, 11], we have the following property of the $\tau \circ \pi$ linear part : some $\tau \circ \pi$ input values equal to zero except on at most four bytes are transformed into output values equal to zero except for at most four bytes. M. Minier and H. Gilbert [11] proposed two sets of possible input values which satisfy this property and showed that this property, as applied to difference values in the encryption of pairs of plaintexts, enables us to make many iterative truncated differentials. To compute a lower bound on probabilities of iterative truncated

Table 1. Comparison of our results with the previous attacks on CRYPTON

Attack	Version	Rounds	Data (CP)	Time (Enc)
Square Attack [6]	$V0.5$, $V1.0$	6-round	$2^{32}(2^{32}$ Mem)	2^{56}
Impossible Differential Attack [13]	$V0.5$, $V1.0$	5-round	$2^{83.4}$	2^{43}
Impossible Differential Attack [3]	$V0.5$, $V1.0$	6-round	2^{91}	2^{124}
Stochastic Attack [11]	$V0.5$	8-round	$2^{114.6}$	$2^{114.6}$
			2^{112}	2^{112}
Truncated Differential Attack (This paper)	$V1.0$	7-round	$2^{97}(2^{39}$ Mem)	$2^{126.2}$
			$2^{97}(2^{100}$ Mem)	$2^{97.2}$
		8-round	$2^{126}(2^{100}$ Mem)	$2^{126.2}$

$V0.5$: the initial version of CRYPTON, $V1.0$: the modified version of CRYPTON
CP : Chosen Plaintexts, Enc : Encryption units, Mem : Memory

differentials M. Minier and H. Gilbert introduced transition probabilities matrices. In this paper, we show how to exploit the matrices to improve the lower bound proposed by M. Minier and H. Gilbert. Based on our computer experiments, we found a lower bound $2^{-93.1}$ on probabilities of 4-round differentials, and a lower bound $2^{-116.9}$ on probabilities of 5-round differentials. We exploit the 4-round differential to attack 7-round CRYPTON with success rate 75% and the 5-round differential to attack 8-round CRYPTON with success rate 99%. See Table 1 for a summary of results presented in this paper and their comparison with the previous attacks.

This paper is organized as follows: Section 2 provides a short description of CRYPTON and Section 3 shows how to compute a lower bound on probabilities of iterative truncated differentials. In Section 4, we design our truncated differential attacks on CRYPTON. Finally, Section 5 concludes the paper.

2 Description of CRYPTON

The 128-bit block cipher CRYPTON [9] is a 12-round substitution-permutation network (SPN) with a key length of up to 256 bits. We represent a 128-bit data in a 4×4 byte array as follows.

$$A = (A[3], A[2], A[1], A[0])^t = \begin{pmatrix} A[0] \\ A[1] \\ A[2] \\ A[3] \end{pmatrix} = \begin{pmatrix} a_{0,3} & a_{0,2} & a_{0,1} & a_{0,0} \\ a_{1,3} & a_{1,2} & a_{1,1} & a_{1,0} \\ a_{2,3} & a_{2,2} & a_{2,1} & a_{2,0} \\ a_{3,3} & a_{3,2} & a_{3,1} & a_{3,0} \end{pmatrix}$$

One round of CRYPTON consists of applying γ, π, τ and σ in sequence to the 4×4 byte array. More specifically, the odd round function ρ_{o_K} and the even round function ρ_{e_K} are defined (for the round key K) by

$$\rho_{o_K} = \sigma_K \circ \tau \circ \pi_o \circ \gamma_o \text{ for odd rounds,}$$
$$\rho_{e_K} = \sigma_K \circ \tau \circ \pi_e \circ \gamma_e \text{ for even rounds,}$$

S_3	S_2	S_1	S_0
S_0	S_3	S_2	S_1
S_1	S_0	S_3	S_2
S_2	S_1	S_0	S_3

γ_o

S_1	S_0	S_3	S_2
S_2	S_1	S_0	S_3
S_3	S_2	S_1	S_0
S_0	S_3	S_2	S_1

γ_e

Fig. 1. Description of the substitution γ

where $(f \circ g)(x) = f(g(x))$.

- γ_o and γ_e are nonlinear byte-wise substitutions that consist of four S-boxes each. The four S-boxes are derived from S as follows: for each $x \in \{0, 1, \cdots, 255\}$,

$$S_0(x) = S(x)^{\lll 1}, \quad S_1(x) = S(x)^{\lll 3}, \quad S_2(x) = S(x^{\ggg 1}), \quad S_3(x) = S(x^{\ggg 3}),$$

where $X^{\lll n}$ (resp. $X^{\ggg n}$) represents rotation to the left (resp. right) by n bits. γ_o and γ_e are represented as in Figure 1. (See [9] for the details of S.)

- π_o and π_e are linear bit permutations that consist of four column-wise bit permutations π_0, π_1, π_2 and π_3 each. If the input and output columns of π_0 are $(a_3, a_2, a_1, a_0)^t$ and $(b_3, b_2, b_1, b_0)^t$, respectively, π_0 is defined by

$$\begin{aligned}
b_0 &\leftarrow (a_3 \oplus m_3) \oplus (a_2 \oplus m_2) \oplus (a_1 \oplus m_1) \oplus (a_0 \oplus m_0)\\
b_1 &\leftarrow (a_3 \oplus m_0) \oplus (a_2 \oplus m_3) \oplus (a_1 \oplus m_2) \oplus (a_0 \oplus m_1)\\
b_2 &\leftarrow (a_3 \oplus m_1) \oplus (a_2 \oplus m_0) \oplus (a_1 \oplus m_3) \oplus (a_0 \oplus m_2)\\
b_3 &\leftarrow (a_3 \oplus m_2) \oplus (a_2 \oplus m_1) \oplus (a_1 \oplus m_0) \oplus (a_0 \oplus m_3)
\end{aligned}$$

where $m_0 = fc_x,\ m_1 = f3_x,\ m_2 = cf_x$ and $m_3 = 3f_x$. π_1, π_2 and π_3 are derived from π_0 as follows.

$$\begin{pmatrix} b_0\\ b_1\\ b_2\\ b_3 \end{pmatrix} = \pi_0 \begin{pmatrix} a_0\\ a_1\\ a_2\\ a_3 \end{pmatrix} \Rightarrow \begin{pmatrix} b_1\\ b_2\\ b_3\\ b_0 \end{pmatrix} = \pi_1 \begin{pmatrix} a_0\\ a_1\\ a_2\\ a_3 \end{pmatrix},$$

$$\begin{pmatrix} b_2\\ b_3\\ b_0\\ b_1 \end{pmatrix} = \pi_2 \begin{pmatrix} a_0\\ a_1\\ a_2\\ a_3 \end{pmatrix} \quad , \quad \begin{pmatrix} b_3\\ b_0\\ b_1\\ b_2 \end{pmatrix} = \pi_3 \begin{pmatrix} a_0\\ a_1\\ a_2\\ a_3 \end{pmatrix}$$

Then π_o and π_e are represented by $(\pi_3, \pi_2, \pi_1, \pi_0)$ and $(\pi_1, \pi_0, \pi_3, \pi_2)$, respectively.

- τ is a linear transposition. It simply moves the byte at $a_{i,j}$ to $a_{j,i}$.

- σ_K is a bite-wise key XOR.

The block cipher CRYPTON can be described as

$$\phi_e \circ \rho_{e_{K^{12}}} \circ \rho_{o_{K^{11}}} \cdots \rho_{e_{K^2}} \circ \rho_{o_{K^1}} \circ \sigma_{K^0}$$

where the linear output transformation $\phi_e = \tau \circ \pi_e \circ \tau$ is used at the last round. The block cipher CRYPTON can be also described as

$$\phi_e \circ \rho^{eq}_{e_{K^{12}}} \circ \rho^{eq}_{o_{K^{11}}} \cdots \rho^{eq}_{e_{K^2}} \circ \rho^{eq}_{o_{K^1}} \circ \sigma_{K^0}$$

where $\rho^{eq}_{o_{K^i}} = \tau \circ \pi_o \circ \sigma_{(\pi_o^{-1} \circ \tau^{-1})(K^i)} \circ \gamma_o$ for odd rounds and $\rho^{eq}_{e_{K^i}} = \tau \circ \pi_e \circ \sigma_{(\pi_e^{-1} \circ \tau^{-1})(K^i)} \circ \gamma_e$ for even rounds. We denote $(\pi_o^{-1} \circ \tau^{-1})(K^i)$ and $(\pi_e^{-1} \circ \tau^{-1})(K^i)$ by K^i_{eq}.

3 Truncated Differentials of CRYPTON

In this Section, we study the way to exploit an iterative property of the $\tau \circ \pi$ linear part to make truncated differentials with relatively high probabilities. (This method was already introduced by M. Minier and H. Gilbert [11].) And we show how to improve the lower bound on probabilities of truncated differentials proposed by M. Minier and H. Gilbert.

Property 1. ([11]) Assume that two sets D_1 and D_2 are defined as follows.

$$D_1 = \{01_x, 02_x, 03_x, 10_x, 11_x, 12_x, 13_x, 20_x, 21_x, 22_x, 23_x, 30_x, 31_x, 32_x, 33_x\},$$
$$D_2 = \{04_x, 08_x, 0c_x, 40_x, 44_x, 48_x, 4c_x, 80_x, 84_x, 88_x, 8c_x, c0_x, c4_x, c8_x, cc_x\}.$$

Then there exist the following eight equations related to D_1, D_2 and $\tau \circ \pi$. ($\delta_1, \delta_2 \in D_1$, $\delta_3, \delta_4 \in D_2$, π: π_o or π_e)

$$\tau \circ \pi \begin{pmatrix} \delta_1 & 0 & \delta_2 & 0 \\ 0 & 0 & 0 & 0 \\ \delta_1 & 0 & \delta_2 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \delta_2 & 0 & \delta_2 & 0 \\ 0 & 0 & 0 & 0 \\ \delta_1 & 0 & \delta_1 & 0 \end{pmatrix}, \quad \tau \circ \pi \begin{pmatrix} \delta_3 & 0 & \delta_4 & 0 \\ 0 & 0 & 0 & 0 \\ \delta_3 & 0 & \delta_4 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \delta_4 & 0 & \delta_4 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_3 & 0 & \delta_3 \end{pmatrix}$$

$$\tau \circ \pi \begin{pmatrix} 0 & \delta_1 & 0 & \delta_2 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & \delta_2 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \delta_2 & 0 & \delta_2 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & \delta_1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \tau \circ \pi \begin{pmatrix} 0 & \delta_3 & 0 & \delta_4 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_3 & 0 & \delta_4 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \delta_4 & 0 & \delta_4 & 0 \\ 0 & 0 & 0 & 0 \\ \delta_3 & 0 & \delta_3 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\tau \circ \pi \begin{pmatrix} 0 & 0 & 0 & 0 \\ \delta_1 & 0 & \delta_2 & 0 \\ 0 & 0 & 0 & 0 \\ \delta_1 & 0 & \delta_2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \delta_2 & 0 & \delta_2 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & \delta_1 \end{pmatrix}, \quad \tau \circ \pi \begin{pmatrix} 0 & 0 & 0 & 0 \\ \delta_3 & 0 & \delta_4 & 0 \\ 0 & 0 & 0 & 0 \\ \delta_3 & 0 & \delta_4 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \delta_4 & 0 & \delta_4 & 0 \\ 0 & 0 & 0 & 0 \\ \delta_3 & 0 & \delta_3 & 0 \end{pmatrix}$$

$$\tau \circ \pi \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & \delta_2 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & \delta_2 \end{pmatrix} = \begin{pmatrix} \delta_2 & 0 & \delta_2 & 0 \\ 0 & 0 & 0 & 0 \\ \delta_1 & 0 & \delta_1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \tau \circ \pi \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \delta_3 & 0 & \delta_4 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_3 & 0 & \delta_4 \end{pmatrix} = \begin{pmatrix} 0 & \delta_4 & 0 & \delta_4 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_3 & 0 & \delta_3 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Since an output difference of $\tau \circ \pi$ is an input difference of the γ nonlinear part of the next round transformation, one might have the following input and output differences of γ.

$$\gamma \begin{pmatrix} 0 & \delta_j & 0 & \delta_j \\ 0 & 0 & 0 & 0 \\ 0 & \delta_i & 0 & \delta_i \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \delta'_i & 0 & \delta'_j \\ 0 & 0 & 0 & 0 \\ 0 & \delta'_i & 0 & \delta'_j \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (1)$$

where ($\delta_i, \delta_j \in D_1$ or $\delta_i, \delta_j \in D_2$) and ($\delta'_i, \delta'_j \in D_1$ or $\delta'_i, \delta'_j \in D_2$). If this is the case of odd round, the probability p that satisfies Equation (1) can be computed as follows.

$$p = DP(\delta_j \overset{S_2}{\rightarrow} \delta'_i) \cdot DP(\delta_j \overset{S_0}{\rightarrow} \delta'_j) \cdot DP(\delta_i \overset{S_0}{\rightarrow} \delta'_i) \cdot DP(\delta_i \overset{S_2}{\rightarrow} \delta'_j)$$

where $DP(\delta \overset{S_i}{\rightarrow} \delta')$ represents the probability of getting a δ' output difference from a δ input difference after the S_i-box. We denote the probability p by $Pr((\delta_i, \delta_j) \rightarrow (\delta'_i, \delta'_j))$. As like Equation (1), all of the output differences of $\tau \circ \pi$ in Property 1 can be applied to γ of the next round transformation, and an output difference of γ is an input difference of $\tau \circ \pi$. Thus we can make many iterative truncated differentials over several rounds by applying the above process to consecutive rounds iteratively.

In this paragraph, we deal with transition probabilities matrices [11] which represent all probabilities of transition between input differences (δ_i, δ_j) and output differences (δ'_i, δ'_j) of γ. Since the two sets D_1 and D_2 have 15 elements each, we obtain a 225×225 matrix whose columns correspond to input differences (δ_i, δ_j) and rows correspond to output differences (δ'_i, δ'_j) and entries are defined by $Pr((\delta_i, \delta_j) \rightarrow (\delta'_i, \delta'_j))$. If we take into account the eight equations of Property 1 and the forms of input and output differences of γ, then we have total 32 225×225 matrices, denoted M^{xy}_{mn} where $m, n \in \{1, 2\}$, $x \in \{o, e\}$ and $y \in \{a, b, c, d\}$. Here, m (resp. n) represents the set D_1 or D_2 of input differences (resp. output differences) of γ, x represents odd round (o) or even round (e), and y represents types of input and output differences of γ. Type a means the difference pattern which is zero at all bytes but $(0, 3)$, $(0, 1)$, $(2, 3)$ and $(2, 1)$, Type b means the difference pattern which is zero at all bytes but $(0, 2)$, $(0, 0)$, $(2, 2)$ and $(2, 0)$, Type c means the difference pattern which is zero at all bytes but $(1, 3)$, $(1, 1)$, $(3, 3)$ and $(3, 1)$, and Type d means the difference pattern which is zero at all bytes but $(1, 2)$, $(1, 0)$, $(3, 2)$ and $(3, 0)$. (For example, Equation (1) belongs to Type b.) Since γ_o and γ_e use the four S-boxes repeatedly as like Figure 1, it holds that $M^{oa}_{mn} = M^{od}_{mn}$, $M^{ea}_{mn} = M^{ed}_{mn}$, $M^{oc}_{mn} = M^{eb}_{mn}$ and $M^{ob}_{mn} = M^{ec}_{mn}$ where $m, n \in \{1, 2\}$. Thus we can reduce the number of 225×225 matrices which are considered in our analysis by half.

M. Minier and H. Gilbert [11] showed that a 225×225 matrix obtained by the multiplication of n transition probabilities matrices represents key-independent transition probabilities between input and output differences on an n rounds scheme. (We denote the obtained matrix on an n rounds scheme by n-round

Table 2. Computation of 4- and 5-round truncated differential probabilities

	Path	4-round Transition Probabilities Matrix	Prob.
Input difference : (cc_x, cc_x) Output difference : Type b	22112	$M_{12}^{od} \cdot M_{11}^{ec} \cdot M_{21}^{oa} \cdot M_{22}^{eb}$	$\frac{36.2}{2^{103}}$
	22221	$M_{21}^{ob} \cdot M_{22}^{ed} \cdot M_{22}^{oa} \cdot M_{22}^{eb}$	$\frac{30.8}{2^{103}}$
	21111	$M_{11}^{ob} \cdot M_{11}^{eb} \cdot M_{11}^{ob} \cdot M_{21}^{eb}$	$\frac{794.2}{2^{103}}$
	21222	$M_{22}^{od} \cdot M_{22}^{ea} \cdot M_{12}^{ob} \cdot M_{21}^{eb}$	$\frac{88.0}{2^{103}}$
Sum of the 4-round differential probabilities			$2^{-93.1}$

	Path	5-round Transition Probabilities Matrix	Prob.
Input difference : $(c4_x, c4_x)$ Output difference : Type a	221122	$M_{22}^{eb} \cdot M_{12}^{od} \cdot M_{11}^{ec} \cdot M_{21}^{oa} \cdot M_{22}^{eb}$	$\frac{131.8}{2^{128}}$
	222212	$M_{12}^{eb} \cdot M_{21}^{ob} \cdot M_{22}^{ed} \cdot M_{22}^{oa} \cdot M_{22}^{eb}$	$\frac{362.2}{2^{128}}$
	211112	$M_{12}^{eb} \cdot M_{11}^{ob} \cdot M_{11}^{eb} \cdot M_{11}^{ob} \cdot M_{21}^{eb}$	$\frac{74.0}{2^{128}}$
	212222	$M_{22}^{eb} \cdot M_{22}^{od} \cdot M_{22}^{ea} \cdot M_{12}^{ob} \cdot M_{21}^{eb}$	$\frac{338.3}{2^{128}}$
	221211	$M_{11}^{ed} \cdot M_{21}^{oc} \cdot M_{12}^{ec} \cdot M_{21}^{oa} \cdot M_{22}^{eb}$	$\frac{379.6}{2^{128}}$
	222121	$M_{21}^{ed} \cdot M_{12}^{oa} \cdot M_{21}^{ed} \cdot M_{22}^{oa} \cdot M_{22}^{eb}$	$\frac{171.2}{2^{128}}$
	211221	$M_{21}^{ed} \cdot M_{22}^{oa} \cdot M_{12}^{eb} \cdot M_{11}^{ob} \cdot M_{21}^{eb}$	$\frac{15.2}{2^{128}}$
	212111	$M_{11}^{ed} \cdot M_{11}^{oc} \cdot M_{21}^{ea} \cdot M_{12}^{ob} \cdot M_{21}^{eb}$	$\frac{709.4}{2^{128}}$
Sum of the 5-round differential probabilities			$2^{-116.9}$

transition probabilities matrix.) For example, a 2-round transition probabilities matrix $M_{22}^{ea} \cdot M_{12}^{ob}$ represents key-independent transition probabilities between input differences $(\delta_1, \delta_2) \in D_1 \times D_1$ and output differences $(\delta_3, \delta_4) \in D_2 \times D_2$ on 2 rounds scheme. (Note that the round associated with M_{12}^{ob} is located before the round associated with M_{22}^{ea} and the choice of Type a, b, c and d in transition probabilities matrices should be cautious.) A lower bound on the probabilities of this 2-round differentials is provided by summing up the probabilities of all intermediate values which belong to D_2.

M. Minier and H. Gilbert [11] exploited the best column of one of n-round transition probabilities matrices to compute a lower bound of n-round truncated differential probabilities, while we take into account various n-round transition probabilities matrices to compute it. (The best column represents the one that the sum of its entries is larger than those of other columns.) That is, increasing the number of n-round transition probabilities matrices which can be used in our attack (i.e. the output differences associated with the n-round transition probabilities matrices belong to the same type.), we improve the lower bound suggested by M. Minier and H. Gilbert. For 6-round truncated differentials, we can increase the lower bound $2^{-151.23}$ suggested by M. Minier and H. Gilbert up to $2^{-147.9}$.

Based on our computer experiments,[1] we obtain a 4-round truncated differential with probability $2^{-93.1}$ for rounds 2-5 (Round indexes start with 1). That is,

[1] We carried out the computer experiments through two steps, i.e., the first step is for storing the 16 transition probabilities matrices and the second step is for obtaining the n-round transition probabilities matrices. The former step is performed by using

if the input difference is the (cc_x, cc_x) pair of Type b, the probability that the output difference be a $(\delta'_i, \delta'_j)(\in D_1 \times D_1$ or $D_2 \times D_2)$ pair of Type b is $2^{-93.1}$. We also obtain a 5-round truncated differential with probability $2^{-116.9}$ for rounds 2-6. That is, if the input difference is the $(c4_x, c4_x)$ pair of Type b, the probability that the output difference be a $(\delta'_i, \delta'_j)(\in D_1 \times D_1$ or $D_2 \times D_2)$ pair of Type a is $2^{-116.9}$. Table 2 illustrates computation of the 4- and 5-round truncated differential probabilities. In Table 2, Path means the path of the γ nonlinear part in each round (for example, Path 22112 in Table 2 represents $D_2 \xrightarrow{\gamma_e} D_2 \xrightarrow{\gamma_o} D_1 \xrightarrow{\gamma_e} D_1 \xrightarrow{\gamma_o} D_2$), and Prob. means the sum of probabilities in the column associated with (cc_x, cc_x) for 4 rounds scheme (or $(c4_x, c4_x)$ for 5 rounds scheme).

Actually, the truncated differentials proposed by M. Minier and H. Gilbert are a bit different from ours. Our differentials have output differences whose cardinality is exactly $455(= |D_1 \times D_1| + |D_2 \times D_2|)$, while the truncated differentials proposed by M. Minier and H. Gilbert have output differences whose cardinality is exactly $225(= |D_i \times D_i|, i \in \{1, 2\})$. Even though our differentials offer a smaller fraction of filtering wrong pairs out than those of M. Minier and H. Gilbert, they have higher probabilities than those of M. Minier and H. Gilbert. This fact allows us to retrieve subkey material with less data complexity and the same success rate.

4 Attacks on CRYPTON with a 128-bit Key

We present here attacks on 7 and 8-round CRYPTON with the first addition of subkey σ_{K_0} and the final transformation ϕ_e. We introduce Algorithm 1 for attacking 7-round CRYPTON, and then improve this Algorithm 1 by using difference distribution tables for the S-boxes. We denote the improved algorithm by Algorithm 2. Using Algorithm 2 we can attack the CRYPTON cipher up to 8 out of its 12 rounds. Algorithm 1 for attacking 7-round CRYPTON is performed as follows.

1. Prepare 2^{65} pools of 2^{32} plaintexts P_i^j, $j = 0, \cdots, 2^{65} - 1$, $i = 0, \cdots, 2^{32} - 1$ that have all possible values in bytes $(1,2)$,$(1,0)$,$(3,2)$ and $(3,0)$ and are constant in the other bytes. Encrypt the pools to get 2^{65} pools of 2^{32} ciphertexts C_i^j.
2. Decrypt the 2^{65} pools C_i^j through $\pi_o^{-1} \circ \pi_e^{-1} \circ \tau^{-1} (= \pi_o^{-1} \circ \tau^{-1} \circ \tau^{-1} \circ \pi_e^{-1} \circ \tau^{-1})$, and denote the 2^{65} pools we get by C_i^{j*} (each C_i^{j*} represents a 128-bit data after the $\sigma_{K_{eq}^7}$ key addition part in round 7).
3. Initialize an array of 2^{128} counters corresponding to possible subkeys for $(K^0_{1,2}, K^0_{1,0}, K^0_{3,2}, K^0_{3,0}, K^6_{eq,0,2}, K^6_{eq,0,0}, K^6_{eq,2,2}, K^6_{eq,2,0}, K^7_{eq,0,3}, K^7_{eq,0,2}, K^7_{eq,0,1}, K^7_{eq,0,0}, K^7_{eq,2,3}, K^7_{eq,2,2}, K^7_{eq,2,1}, K^7_{eq,2,0})$.
4. For each 32-bit subkey value K^0 do the following:

the difference distribution tables of the four S-boxes, and the latter step is performed by using the multiplication of n related transition probabilities matrices.

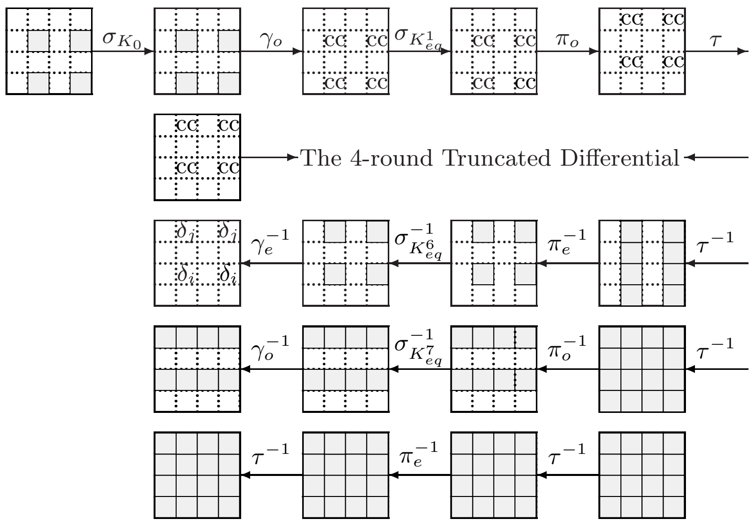


Fig. 2. Truncated Differential Attack on 7-round CRYPTON

(a) Partially encrypt P_i^j through the 4 active S-boxes in round 1, and denote the values we get by P_i^{j*} (each P_i^{j*} represents a 128-bit data after the γ_o nonlinear part in round 1), and sort each pool on bytes $(1,2),(1,0),(3,2)$ and $(3,0)$ and pick the plaintext pairs $(P_{i_1}^j, P_{i_2}^j)$ such that the differences between $P_{i_1}^{j*}$ and $P_{i_2}^{j*}$ are of (cc_x, cc_x) of Type d.

(b) Check that $C_{i_1}^{j*}$ and $C_{i_2}^{j*}$ have zero difference at the rows 1 and 3.

(c) For each 64-bit subkey value K_{eq}^7 do the following:

i. Partially decrypt the pairs $(C_{i_1}^{j*}, C_{i_2}^{j*})$ which passed the above test through $\pi_e^{-1} \circ \tau^{-1} \circ \gamma_o^{-1} \circ \sigma^{-1}_{K^7_{eq}}$ in the 8 active bytes, and denote the values we get by C_i^{j**} (each C_i^{j**} represents a 128-bit data after the $\sigma_{K^6_{eq}}$ key addition part in round 6).

ii. Check that $C_{i_1}^{j**}$ and $C_{i_2}^{j**}$ have zero difference in bytes $(1,2),(1,0)$, $(3,2)$ and $(3,0)$.

iii. For each 32-bit subkey value K_{eq}^6, partially decrypt the pairs $(C_{i_1}^{j**}$, $C_{i_2}^{j**})$ which passed the last test through γ_e^{-1} in the 4 active S-boxes, and check that the differences between the decrypted two texts is of (δ_i, δ_j) of Type b where $(\delta_i, \delta_j) \in D_1 \times D_1$ or $(\delta_i, \delta_j) \in D_2 \times D_2$. If so, increase the corresponding counter by 1.

5. Check all counters, and output the subkey whose counter is greater than or equal to 6.

The data complexity of Algorithm 1 is 2^{97} chosen plaintexts. Due to the 2^{97} chosen plaintexts and 2^{128} counters for subkeys, Algorithm 1 seems to require a number of bytes of memory. However, if we handle Algorithm 1 delicately, the amount of the required memory can be reduced to $2^{39}(= 2^{32} \cdot 2 \cdot 16)$ bytes.

The time complexity of Step 1 (the data collecting step) is 2^{97} 7-round CRYPTON encryptions. The time complexity of Step 2 is $2^{94.2}(= 2^{97} \cdot \frac{1}{7})$ 7-round CRYPTON encryptions. In Step 4-(a), we get about 2^{96} $(P^j_{i_1},P^j_{i_2})$ pairs (which will be tested in the next step) for each 32-bit subkey value K^0. Since Step 4-(b) has a 64-bit condition, the expected number of $(C^{j*}_{i_1},C^{j*}_{i_2})$ pairs which pass this test is about $2^{32}(= 2^{96} \cdot 2^{-64})$. So the time complexity of Step 4-(c)-i is $2^{125.2}(= 2^{32} \cdot 2^{64} \cdot (2 \cdot 2^{32} \cdot \frac{1}{7} \cdot \frac{1}{2}))$ 7-round CRYPTON encryptions. Similarly, after Step 4-(c)-ii, only about one pair remains for each 96-bit subkey value K^0, K^7_{eq} since we have a 32-bit condition on 2^{32} pairs. So the time complexity of Step 4-(c)-iii is $2^{125.2}(= 2^{32} \cdot 2^{64} \cdot 2^{32} \cdot (2 \cdot 1 \cdot \frac{1}{7} \cdot \frac{1}{4}))$ 7-round CRYPTON encryptions. Thus the time complexity of Algorithm 1 is about $2^{126.2}(= 2^{97} + 2^{94.2} + 2^{125.2} + 2^{125.2})$ 7-round CRYPTON encryptions.

For the 2^{65} pools, the expectation of counter of right subkey is about $2^{2.9}(= 2^{96} \cdot 2^{-93.1})$, while the expectation of counter of each wrong subkey is about $2^{-23.2}(= 2^{96} \cdot \frac{2 \cdot 15 \cdot 15}{2^{128}} = 2^{96} \cdot 2^{-119.2})$ (the value $2 \cdot 15 \cdot 15$ represents the number of $(\delta_i, \delta_j)(\in D_1 \times D_1$ or $D_2 \times D_2))$. It follows that the probability that the counter of right key is greater than or equal to 6 is about $0.75(\approx \sum_{i=6}^{\infty}({}_{2^{96}}C_i \cdot (2^{-93.1})^i \cdot (1 - 2^{-93.1})^{2^{96}-i}))$, and the probability that the counter of each wrong subkey is greater than or equal to 6 is about $2^{-148.7}(\approx \sum_{i=6}^{\infty}({}_{2^{96}}C_i \cdot (2^{-119.2})^i \cdot (1 - 2^{-119.2})^{2^{96}-i}))$. Thus, the probability which a wrong subkey is suggested together with the right subkey is about $2^{-20.7}(= 2^{128} \cdot 2^{-148.7})$, and the success rate of Algorithm 1 is about $0.75(\approx 0.75 \cdot (\sum_{i=0}^{5}({}_{2^{96}}C_i \cdot (2^{-119.2})^i \cdot (1 - 2^{-119.2})^{2^{96}-i}))^{2^{128}})$.

Now we describe Algorithm 2 which improves Algorithm 1 by adding precomputation step. Algorithm 2 for attacking 7-round CRYPTON is performed as follows.

Precomputation Step : Prepare difference distribution tables for the four S-boxes and store the possible pairs of each of their entries in a special preprocessed table, denoted Table A, and prepare another large special preprocessed table, denoted Table B, associated with input and output differences of γ in rounds 6 and 7. To make Table B we first compute 2^{32} (input value, output value) pairs of $\tau \circ \pi_e$ where input values of $\tau \circ \pi_e$ have all possible values in bytes $(0, 2), (0, 0), (2, 2)$ and $(2, 0)$ and are zero in the other bytes. (Note that the input values of $\tau \circ \pi_e$ represent the output differences of γ_e in round 6, and the output values of $\tau \circ \pi_e$ represent the input differences of γ_o in round 7.) Table B consists of a $2^{32} \times 2^{64}$ array whose rows correspond to (input value, output value) pairs of $\tau \circ \pi_e$ in round 6, and columns correspond to the possible output differences of γ_o in round 7 (i.e. columns correspond to all possible differences at the rows 0 and 2). If an entry of the $2^{32} \times 2^{64}$ array is assigned by a (I, O) pair of $\tau \circ \pi_e$ and the output difference Δ of γ_o in round 7, we store in this entry all possible output pairs which satisfy $(\delta_1, \delta_2) \xrightarrow{\gamma_e} I$ in Table A where $(\delta_1, \delta_2)(\in D_1 \times D_1$ or $D_2 \times D_2)$ represent any differences of Type b, together with all possible output pairs which satisfy $O \xrightarrow{\gamma_e} \Delta$ in Table A.

1. Perform Steps 1,2, and 3 of Algorithm 1.
2. From each pool, we generate 2^{63} pairs $(C^{j*}_{i_1}, C^{j*}_{i_2})$. Check that $C^{j*}_{i_1}$ and $C^{j*}_{i_2}$ have zero difference at the rows 1 and 3.
3. For each plaintext pair $(P^{j}_{i_1}, P^{j}_{i_2})$ related to $(C^{j*}_{i_1}, C^{j*}_{i_2})$ which passed the above test, do the following:
   (a) Obtain the values of the 32-bit subkey K^0 by using Table A.
   (b) For each pair $(C^{j*}_{i_1}, C^{j*}_{i_2})$, and for each (input value, output value) of $\tau \circ \pi_e$, do the following:
      i. Obtain the values of the 64-bit subkey K^7_{eq} by using Table B.
      ii. Using the obtained 64-bit subkeys K^7_{eq}, partially decrypt $(C^{j*}_{i_1}, C^{j*}_{i_2})$ through $\pi_e^{-1} \circ \tau^{-1} \circ \gamma_o^{-1} \circ \sigma^{-1}_{K^7_{eq}}$ in the 16 active bytes, and obtain the values of the 32-bit subkey K^6_{eq} by using the decrypted two texts and Table B. Increase the counters related to the obtained 128-bit subkeys by 1.
4. Check all counters, and output the subkey whose counter is greater than or equal to 6.

The amount of the required memory of Algorithm 2 is dominated by Table B. Table B can be divided by two parts of memory, i.e., one is associated with round 6 and the other is associated with round 7. Since the memory requirements of the first part is about $2^{43.3}(= 4 \cdot 2 \cdot (2 \cdot 15 \cdot 15) \cdot 2^{32})$ bytes, and the memory requirements of the second part is about $2^{100}(= 8 \cdot 2 \cdot 2^{32} \cdot 2^{64})$ bytes, the memory requirements of Algorithm 2 is about 2^{100} bytes.

We now analyze the time complexity of Algorithm 2. After Step 2, we expect to have about 2^{64} $(C^{j*}_{i_1}, C^{j*}_{i_2})$ pairs left since we have a 64-bit condition on $2^{128}(= 2^{65} \cdot 2^{63})$ pairs. To obtain the values of the 32-bit subkey K^0 in Step 3-(a) we should look up Table A about 2^{64} times. Similarly, to obtain the values of the 64-bit subkey K^7_{eq} in Step 3-(b)-i we should look up Table B about $2^{96}(= 2^{64} \cdot 2^{32})$ times. Since the expectation of the remained pairs is 2^{64}, the time complexity of Step 3-(b)-ii is about $2^{92.2}(= 2^{64} \cdot 2^{32} \cdot \frac{1}{2} \cdot \frac{1}{7})$ 7-round CRYPTON encryptions. Thus the time complexity of Algorithm 2 is about $2^{97.2}(= 2^{97} + 2^{94.2} + 2^{92.2})$ 7-round CRYPTON encryptions.

Similarly, we can attack 8-round CRYPTON by applying the forgoing 5-round truncated differential to Algorithm 2. The attack on 8-round CRYPTON requires about 2^{126} chosen plaintexts and $2^{126.2}(= 2^{126} + 2^{123.2} + 2^{121.2})$ encryptions. In addition, the attack requires to look up Table A about 2^{93} times and Table B about 2^{125}. If we increases the threshold up to 200, we can increase the success rate of the attack up to 99%.

5 Conclusion

In this paper, we showed how to improve a lower bound on probabilities of truncated differentials proposed by M. Minier and H. Gilbert. The improvement allows us to make a 4-round truncated differential with probability $2^{-93.1}$ and a 5-round truncated differential with probability $2^{-116.9}$. Using these differentials we showed that 7- and 8-round CRYPTON could be breakable. These are the best known attacks for this cipher. See Table 1 for a summary of results presented in this paper and their comparison with previously known attacks.

References

[1] O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, S. Vaudenay, *Report on the AES Candidates*, The Second Advanced Encryption Standard Candidate Conference, N. I. S. T., 1999. 446

[2] E. Biham and A. Shamir, *Differential cryptanalysis of the full 16-round DES*, Advances in Cryptology - CRYPTO'92, LNCS 740, Springer-Verlag, 1992, pp. 487–496.

[3] J. Cheon, M. Kim, K. Kim, and J. Lee, *Improved Impossible Differential Cryptanalysis of Rijndael and Crypton,* ICISC'01, LNCS 2288, Springer-verlag, 2001, pp 39-49. 446, 447

[4] J. Daemen, L. Knudsen, and V. Rijndael, *The block cipher Square*, FSE'97, LNCS 1267, Springer-verlag, 1997, pp 149-171. 446

[5] J. Daemen and V. Rijndael, *The Rijndael block cipher*, AES proposal, 1998.

[6] C. D'Halluin, G. Bijnens, V. Rijmen, and B. Preneel, *Attack on Six Rounds of Crypton*, FSE'99, LNCS 1636, Springer-verlag, 1999, pp 46-59. 446, 447

[7] L. R. Knudsen, *Trucated and Higher Order Differentials*, FSE'96, LNCS 1039, Springer-Verlag, 1996, pp 196-211.

[8] C. Lim, *CRYPTON : A New 128-bit Block Cipher*, AES Proposal, 1998. 446

[9] C. Lim, *A Revised Version of Crypton - Crypton Version 1.0*, FSE'99, LNCS 1638, Springer-verlag, 1999, pp 31-45. 446, 447, 448

[10] M. Matsui, *Linear cryptanalysis method for DES cipher*, Advances in Cryptology - EUROCRYPT'93, LNCS 765, Springer-Verlag, 1994, pp 386-397.

[11] M. Minier and H. Gilbert, *Stochastic Cryptanalysis of Crypton*, FSE'00, LNCS 1978, Springer-verlag, 2000, pp 121-133. 446, 447, 449, 450, 451

[12] K. Nyberg and Lars R. Knudsen, *Provable security against differential cryptanalysis*, Advances in Cryptology - CRYPTO'92, LNCS 740, Springer-Verlag, 1992, pp 566-574.

[13] H. Seki and T. Kaneko, *Cryptanalysis of Five Rounds of CRYPTON Using Impossible Differentials*, Advances in Cryptology - ASIACRYPT'99, LNCS 1716, Springer-verlag, 1999, pp 45-51. 446, 447