Modulus Search for Elliptic Curve Cryptosystems

Kenji Koyama, Yukio Tsuruoka, and Noboru Kunihiro

NTT Communication Science Laboratories 2-4, Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan {koyama, tsuru, kunihiro}@cslab.kecl.ntt.co.jp

Abstract. We propose a mathematical problem, and show how to solve it elegantly. This problem is related with elliptic curve cryptosystems (ECC). The solving methods can be applied to a new paradigm of key generations of the ECC.

1 Problem

Celebrating Asiacrypt'99 held in November (11th month) of 1999, we propose a mathematical problem after these numbers.

Let c, x, y be integers such that $0 \le x < c$ and $0 \le y < c$. Define N(c) be the number of points (x, y) satisfying $y^2 \equiv x^3 + 11x \pmod{c}$ (1) Obtain all values of c such that N(c) = 1999.

2 Solving the Problem

2.1 Observing the Behaviour of N(c)

This problem itself is easy to understand for junior highschool students, however, solving it may be a little difficult for them. It would be moderate for modern cryptographers.

First, observe the behavior of N(c) concerning equation (1) from small numerical examples. When c=7, the integer points (x,y) of equation (1) are (0,0),(2,3),(2,4),(3,2),(3,5),(6,3),(6,4). Thus, we have N(7)=7. Similarly, we compute the values of N(c) for integers c such that $1 \le c \le 18$, and primes below 100. The result is shown in Table 1.

Table 1. Examples of N(c)

\overline{c}																		
N(c)	1	2	3	6	3	6	7	12	9	6	11	18	17	14	9	24	9	18

c																		
N(c)	19	23	25	31	39	33	43	47	39	59	49	67	71	57	79	83	79	115

We can find the properties of N(c) if we observe Table 1 carefully.

2.2 Obtaining one Solution

Hereafter, considering equation (1) as general as possible, we can obtain the following theorems for the properties of N(c).

Theorem A: Define $N_u(c)$ be the number of points for a general congruence:

$$f(x,y) \equiv 0 \pmod{c} \tag{1}$$

If c is composite (i.e. not prime), then $N_u(c)$ is composite. In particular, when c_1 and c_2 are coprime, we have

$$N_u(c_1c_2) = N_u(c_1)N_u(c_2). (2)$$

Of course, Theorem A holds for N(c) concerning equation (1). If c is a prime power, we have Theorem B.

Theorem B: Define $N_s(c)$ be the number of points for congruence:

$$y^2 \equiv x^3 + ax \pmod{c} \tag{3}$$

Let p be a prime and $c = p^n \ (n \ge 2)$.

(i) If $p \neq 2$ is coprime to a, then $N_s(p^n) = p^{n-1} \cdot N_s(p)$,

(ii) If $p \neq 2$ divides a, then $N_s(p^n) = (2p-1) \cdot N_s(p)^{n-1}$.

(iii) $N_s(2) = 2$. When a = 11, $N_s(2^n) = 3 \cdot 2^{n-1}$.

Putting a = 11 in equation (4), Theorem B holds for N(c) concerning equation (1).

Theorem C: Define $N_s(c)$ in the same way as Theorem B. If p is a prime such that $p \equiv 3 \pmod{4}$ and a is coprime to p, then $N_s(p) = p$.

Theorem C holds for N(c) concerning equation (1).

In the problem, we just said on purpose "c is an integer." I did not say "c is restricted to a prime." Theorem A can be rewritten as "If N(c) is a prime, then c is a prime." Note that 1999 is a prime. We can observe that c is a prime because N(c) = 1999. Moreover, noticing $1999 \equiv 3 \pmod{4}$, we can find from Theorem C that N(1999) = 1999. That is, a prime c satisfying N(c) = 1999 and $c \equiv 3 \pmod{4}$ is only 1999.

2.3 Obtaining other Solutions

In the problem, we said "Obtain all values of c." Therefore, the remaining candidates of c must be primes with $c \equiv 1 \pmod{4}$. What is the range for searching the remaining prime candidates? We show here a strong theorem, which is called Hasse's Theorem and popular in elliptic curve theory.

Hasse's Theorem: Let p be a prime and coprime to $4a^3 + 27b^2$. Consider an elliptic curve over prime field GF(p):

$$y^2 \equiv x^3 + ax + b \pmod{p}. \tag{4}$$

Excluding a point at infinity, the number of points on this curve, denoted by $N_w(p)$, is given by in the following range:

$$p - 2\sqrt{p} \le N_w(p) \le p + 2\sqrt{p}. \tag{5}$$

If $4a^3 + 27b^2$ is coprime to p, the curve of equation (5) becomes an elliptic curve, which is a cubic curve without singular points. Equation (5) is called Weierstraß form. Putting a = 11, b = 0, equation (5) becomes equation (1). The number of points on elliptic curves is usually called an order, including one point at infinity. Thus, the order is expressed as $N_w(p) + 1$. In the problem, to avoid difficulty of understanding of a point at infinity, we define N(c) ($N_u(c)$, $N_s(c)$, and $N_w(c)$) excluding a point at infinity. Even for researchers familiar with elliptic curves and their orders, the proposed problem must be a new application paradigm, in which a modulus is determined from given an order of elliptic curve.

Naive Method: Method 1 First, to restrict the range of the solutions c of the problem, we need to get a lemma of Hasse's theorem. Given $N_w(p) (= N(p))$, a prime modulus p of elliptic curve is restricted between a certain range. This range is obtained from equation (5). By rewriting equation (5), we have

$$p^{2} - 2(N_{w}(p) + 2)p + N_{w}(p)^{2} \le 0.$$

By solving p for this quadratic form, we have an inequality:

$$N_w(p) + 2 - 2\sqrt{N_w(p) + 1} \le p \le N_w(p) + 2 + 2\sqrt{N_w(p) + 1}$$
(6)

Putting $N(p) = N_w(p) = 1999$, we can get an explicit range as $1911.6 \le p \le 2090.4$. Thus, the values of modulus c should be searched from 1912 to 2090. In this range, there are twelve primes ($\equiv 1 \pmod{4}$) as 1913, 1933,1949, 1973, 1993, 1997, 2017, 2029, 2053, 2069, 2081 and 2089. The most naive method is to compute N(p) for all of these twelve values of p, and check whether N(p) = 1999. We can find that only p = 2017 satisfies N(p) = 1999.

Elegant Method: Method 2 Elegant methods can be constructed by decreasing the number of candidates of modulus by a simple analysis. Note that for a prime with $p \equiv 1 \pmod 4$, there are integers U, V (U is odd and V is even) such that

$$p = U^2 + V^2. (7)$$

The values of (U, V) is uniquely determined and easily obtained. In elliptic curve theory, the following theorem D is known,

Theorem D: Let p be a prime satisfying $p \equiv 1 \pmod{4}$ and $p = U^2 + V^2$. If $a \not\equiv 0$ is coprime to p, the number of points of equation (4), denoted by $N_s(p)$, is one of the following four candidates:

$$N_s(p) = p \pm 2U, \quad p \pm 2V \tag{8}$$

Let U'=|p-1999|/2 and $W=p-U'^2$. Observing theorem D, W must be a square to satisfy N(p)=1999 for a prime with $p\equiv 1\pmod 4$. For each p of twelve candidates, the computed values of U' and W are shown in Table 2. Observing Table 2, only four primes such that p=1913,2017,2081 and 2089 imply that W are squares. For these reduced four candidates p, the computed values of N(p) are also shown in Table 2. We can find that only p=2017 satisfies N(p)=1999.

p	1913	1933	1949	1973	1993	1997	2017	2029	2053	2069	2081	2089
U'	43	33	25	13	3	1	9	15	27	35	41	45
W	64	844	1324	1804	1984	1996	1936	1804	1324	844	400	64
N(p)	1929	_	_	_	_	_	1999	_	_	_	2121	2105

Table 2. Reduction of primes p and reduced N(p)

More Elegant Method: Method 3 We would show more elegant and efficient method. If we apply equation (8) and theorem D extendedly, we do not need to know and use Hasse's Theorem and its lemma directly. Note that for prime p with $p \equiv 1 \pmod{4}$, $N_s(p) + 1$ is represented as one of four values:

$$N_s(p) + 1 = (U \pm 1)^2 + V^2, \quad U^2 + (V \pm 1)^2$$

Thus, if given $N_s(p) + 1$ is represented as a sum of two squares as $N_s(p) + 1 = \alpha^2 + \beta^2$ ($\alpha \leq \beta$) then (U, V) is one of $(\alpha \pm 1, \beta)$ and $(\alpha, \beta \pm 1)$. We compute candidates of p from these candidates of (U, V). Then we do primality test for p and check whether N(p) = 1999. The passed p become solutions.

We show the above method concretely. Since $N_s(p)+1=1999+1=2000$, we search (α,β) such that $2000=\alpha^2+\beta^2$, noticing $\alpha \leq \sqrt{2000/2}$. We obtain two pairs $(\alpha,\beta)=(8,44)$, (20,40). From each pair, eight candidates of p_i $(1 \leq i \leq 8)$ can be computed as

$$p_1 = (8+1)^2 + 44^2 = 2017, p_2 = (8-1)^2 + 44^2 = 1985,$$

 $p_3 = 8^2 + (44+1)^2 = 2089, p_4 = 8^2 + (44-1)^2 = 1913,$
 $p_5 = (20+1)^2 + 40^2 = 2041, p_6 = (20-1)^2 + 40^2 = 1981,$
 $p_7 = 20^2 + (40+1)^2 = 2081, p_8 = 20^2 + (40-1)^2 = 1921.$

Among these values, only p_1 , p_3 , p_4 and p_7 are primes. These primes are congruent 1 modulo 4, however, only $p_1 = 2017$ satisfies $N(p_i) = 1999$. This method is more efficient than the method 2 because of less primality tests. It is interesting that four candidates derived by method 3 are the same as four candidates derived by method 2.

Much more Elegant Method: Method 4 Moreover, much more elegant method can be constructed by observing the reduced candidates from another viewpoint. When $p \equiv 1 \pmod{4}$, order S = N(p) + 1 is expressed by

$$S = 4t + 3 + L(11, p),$$

where t is the number of the cases that x^3+11x become quadratic residues modulo p when $1 \le x \le (p-1)/2$. Generally, the Legendre symbol L(d,p) means as follows. L=1 if $d \ne 0$ is a quadratic residue modulo prime p; L=-1 if $d \ne 0$ is a quadratic non-residue modulo prime p; L=0 if d=0. To satisfy S=2000, we need that $S\equiv 0\pmod 4$, and 11 is a quadratic residue modulo p. Among four primes 1913, 2017, 2081, and 2089, only p=2017 satisfies L(11,p)=1. Thus, we compute N(p) for only p=2017, and we verify N(2017)=1999.

Note that method 2 and method 3 require four computations of N(p), however, method 4 requires four computations of Legendre symbols and one computation of N(p). Thus, method 4 is more efficient than methods 2 and 3.

3 Counting Points of the Curves

3.1 General Methods

There are several ways to compute N(c) from c. The most naive method is to count the points (x, y) satisfying equation (1) by varying both x and y from 0 to c-1. The computational complexity is $O(c^2(\log c)^2)$. If c is a prime, we can compute N(c) using Legendre symbol L as

$$N(c) = \sum_{x=0}^{c-1} \{1 + L(x^3 + 11x, c)\}.$$

We call this method the Legendre method. Since the Legendre symbol itself can be computed in $O((\log c)^3)$, computation of N(c) by the Legendre method requires $O(c(\log c)^3)$. It is more efficient than the naive method.

When c is about 4 digits, N(c) can be computed in less than one second on a typical personal computer if the Legendre symbol method is used. When c is about 200 digits, the computation of N(c) is intractable even if the Legendre symbol method is used. For large p, counting the points (i.e. order or $N_w(p)$) on an elliptic curve over prime field GF(p) had been a difficult problem historically. However, Schoof discovered an efficient method in 1985. The implementation is rather complicated, but it runs in polynomial time i.e. in $O((\log p)^8)$. Recently, an improved Schoof method, which is also called Schoof-Elkies-Atkin (SEA) method, is used and it runs in $O((\log p)^6)$. This newest counting method is used in the design of elliptic curve method (ECC). Note that ECC is a public-key cryptosystem, which is the most promising scheme in the next generation of the RSA scheme.

3.2 Special Counting Method for the Problem

Return to the problem. Since equation (1) has a restricted parameters a, b and p, we can compute N(p) analytically and efficiently using Theorem E.

Theorem E: $N_s(p)$ is uniquely determined by

$$N_s(p) = p - \overline{\left(\frac{-a}{\pi}\right)_4} \pi - \left(\frac{-a}{\pi}\right)_4 \overline{\pi},\tag{9}$$

where $p = \pi \overline{\pi}$, and π is Gaussian integer Z[i] $(i = \sqrt{-1})$, and $\pi \equiv 1 \pmod{2+2i}$.

Note that $\left(\frac{-a}{\pi}\right)_4 = \{1, -1, i, -i\}$, and computed as $\left(\frac{-a}{\pi}\right)_4 = (-a)^{(p-1)/4} \mod \pi$.

Using Theorem E, we can easily compute N(p) for each p. For example, when p=2017, we have $p=9^2+44^2$, and $N(2017)=2017-2\times 9=1999$. The computational time on a typical computer using Theorem E is also less than one second.

4 Solution

The above discussion result in a solution of the problem. There are only c=1999 and 2017 satisfying equation (1).

Note that if one try to search them on a computer without any knowledge or any analysis, it need infinite time. The theorems and discussions in this paper convince us that there are only two values of c satisfying equation (1).

5 Viewpoint of ECC Design

In cryptographic design, there are two typical methods for constructing secure elliptic curves for the ECC: the SEA method (the point-counting method based on the improved Schoof algorithm) and the CM(Complex Multiplication) method. **The SEA Method:** The point-counting method computes an order of random curve modulo p until the order satisfies the security. Given parameters (a, b) and prime modulus p of elliptic curve, the improved versions of Schoof algorithm can compute order #E(a, b, p) (= S) in $O((\log p)^6)$. Considering the time of the primality check of p and the security check of p, including their success probability, the computational time for obtaining a suitable triple (S, (a, b), p) based on the improved Schoof algorithm is $O((\log p)^7)$.

The CM method: The CM method chooses a secure order first from modulus p, then builds a curve with that order. Given the prime modulus p of an elliptic curve, the Atkin-Morain algorithm and its variants compute the j-invariant of the curve, and obtains order S and parameters (a,b) satisfying S = #E(a,b,p). They run in $O((\log p)^5)$. Considering the time of the primality check of p and the security check of S, including their success probability, the computational time for obtaining a suitable triple (S,(a,b),p) based on the CM method is $O((\log p)^6)$.

That is, the CM method based on the Atkin-Morain algorithm is more efficient than the point-counting method based on the (improved) Schoof algorithm.

Now, we consider a new approach based on problem G and its solution, which follows. This approach, which we call the *modulus-searching method*, is

in another direction among (S,(a,b),p), and is different from SEA method and CM method.

Problem G: Given order S and parameters (a,b) of an elliptic curve, construct an efficient algorithm for determining a prime modulus p satisfying S = #E(a,b,p), if such p exists.

When the values of a, b and S are arbitrary, we can construct a general algorithm for problem G. We can find a prime p satisfying S = #E(a,b,p) if p exists. The time complexity of this algorithm is $O(\sqrt{S}(\log S)^2)$. This general but simple algorithm is not efficient for large S because there are many candidates for the prime modulus.

Therefore, we focus on the constructions of restricted elliptic curves E(a,b,p) with $\{a \neq 0, b = 0\}$, whose j-invariant is 1728, and $\{a = 0, b \neq 0\}$, whose j-invariant is 0. Note that the Atkin-Morain algorithm excludes these "simple" curves. There are a few studies on the ECC using such curves. If $\{a \neq 0, b = 0 \text{ and } p \equiv 1 \pmod{4}\}$ or $\{a = 0, b \neq 0 \text{ and } p \equiv 1 \pmod{3}\}$, then orders of such curves can be easily computed in $O((\log p)^3)$ by the point-counting method based on complex multiplications over the imaginary quadratic field $\mathbf{Q}(\sqrt{-1})$ or $\mathbf{Q}(\sqrt{-3})$. This "special point-counting algorithm" is faster than the general (improved) Schoof algorithm. This "special point-counting method" constructs a suitable triple (S,(a,b),p) in $O((\log p)^5)$. From the viewpoint of problem 1, however, there have been no concrete proposals or deep discussions of efficient algorithms.

In [1] we proposed efficient algorithms for determining prime modulus p from given order S and parameters (a,b) of elliptic curve $E(a,b,p): y^2 \equiv x^3 + ax + b \pmod{p}$, where $\{a \neq 0, b = 0\}$ or $\{a = 0, b \neq 0\}$. First we choose secure order S from its size. Next we search prime modulus p satisfying S = #E(a,b,p). We can obtain a suitable triple (S,(a,b),p) in polynomial time $O((\log S)^5)$. The proposed approach is faster than the previous approaches based on the Schoof algorithm and the Atkin-Morain algorithm.

References

 K. Koyama, N. Kunihiro and Y. Tsuruoka: "Modulus Searching Methods for Secure Elliptic Curve Cryptosystems", Proc. of 1999 Symposium on Cryptography and Information Security (SCIS 99), pp. 863–868 (1999).

On the Lai-Massey Scheme

Serge Vaudenay*

Ecole Normale Supérieure — CNRS Serge.Vaudenay@ens.fr

Abstract. Constructing a block cipher requires to define a random permutation, which is usually performed by the Feistel scheme and its variants. In this paper we investigate the Lai-Massey scheme which was used in IDEA. We show that we cannot use it "as is" in order to obtain results like Luby-Rackoff Theorem. This can however be done by introducing a simple function which has an orthomorphism property. We also show that this design offers nice decorrelation properties, and we propose a block cipher family called Walnut.

Designing a block cipher requires to build a random permutation from a random key. In most of block cipher constructions, we distinguish two approaches. First we use a fixed network with parallel permutations which are modified at their inputs or outputs by subkey values. This was used for instance in Safer [11] and Square [3]. Second we use the Feistel scheme [4] (or one of its variants) which starts from a random function (see Fig. 1). This was used for instance in DES [1] and Blowfish [14]. The literature gives an extra construction which is not in these categories and which was used in the IDEA cipher [9,8]. It uses a simple scheme which we illustrated on Fig. 2 and which we call the "Lai-Massey scheme" throughout the paper. As for the Feistel scheme, this structure relies on a group structure.

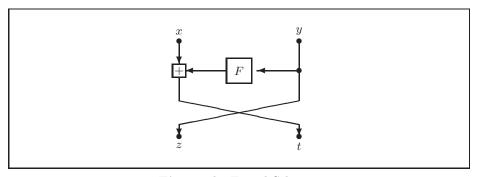


Fig. 1. The Feistel Scheme.

^{*} Part of this work was done while the author was visiting the NTT Laboratories.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 8–19, 1999. © Springer-Verlag Berlin Heidelberg 1999

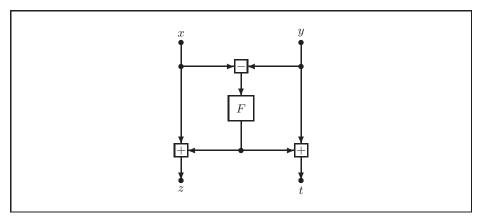


Fig. 2. The Lai-Massey Scheme.

For the Feistel scheme, Luby and Rackoff [10] proved that if the round functions are random, then a 3-round Feistel cipher will look random to any chosen plaintext attack when the number of chosen plaintexts d is negligible towards $2^{\frac{m}{4}}$ (where m is the block length). In this paper, we show a similar result for the Lai-Massey scheme if we add a simple function σ which has the orthomorphism property: it must be such that σ and $x \mapsto \sigma(x) - x$ are both permutations.

The Luby-Rackoff result however holds when the round functions are random. This has been extended by the decorrelation theory [18,19,20,21,22] when the round function have some decorrelation property. This was used to define the Peanut construction family in which the DFC cipher [2,5,6] is an example. We show that we can have similar results with the Lai-Massey scheme and propose a similar construction.

1 Notations

1.1 Feistel and Lai-Massey Schemes

Let (G, +) be a group. Given r functions F_1, \ldots, F_r on G we can define an rround Feistel scheme which is a permutation on G^2 denoted $\Psi(F_1, \ldots, F_r)$. It is
define by iterating the scheme on Fig. 1. If r > 1, we let

$$\Psi(F_1, \dots, F_r)(x, y) = \Psi(F_2, \dots, F_r)(y, x + F_1(y))$$

and

$$\Psi(F_1)(x,y) = (x + F_1(y), y).$$

(The last swap is omitted.)

Similarly, given a permutation σ on G, we define an r-round Lai-Massey scheme as a permutation $\Lambda^{\sigma}(F_1, \ldots, F_r)$ by

$$\Lambda^{\sigma}(F_1,\ldots,F_r)(x,y) = \Lambda^{\sigma}(F_2,\ldots,F_r)(\sigma(x+F(x-y)),y+F(x-y))$$

and

$$\Lambda^{\sigma}(F_1)(x,y) = (x + F(x-y), y + F(x-y))$$

in which the last σ is omitted.

For more convenience, if $x \in G^2$, we let x^l and x^r denote its two halves: $x = (x^l, x^r)$.

1.2 Advantage of Distinguishers and Best Advantage

A distinguisher \mathcal{A} is a probabilistic Turing machine with unlimited computation power. It has access to an oracle \mathcal{O} and can send it a limited number of queries. At the end, the distinguisher must output 0 or 1. We consider the advantage for distinguishing a random function F from a random function G defined by

$$Adv^{\mathcal{A}}(F,G) = |\Pr\left[\mathcal{A}^{\mathcal{O}=F} = 1\right] - |\Pr\left[\mathcal{A}^{\mathcal{O}=G} = 1\right]|.$$

Given an integer d and a random function F from a given set \mathcal{M}_1 to a given set \mathcal{M}_2 , we define the d-wise distribution matrix $[F]^d$ as a matrix in $\mathbf{R}^{\mathcal{M}_1^d \times \mathcal{M}_2^d}$ by

$$[F]_{(x_1,\ldots,x_d),(y_1,\ldots,y_d)}^d = \Pr[F(x_1) = y_1,\ldots,F(x_d) = y_d].$$

For a matrix A in $\mathbf{R}^{\mathcal{M}_1^d \times \mathcal{M}_2^d}$, we define

$$||A||_a = \max_{x_1} \sum_{y_1} \max_{x_2} \sum_{y_2} \dots \max_{x_d} \sum_{y_d} |A_{(x_1,\dots,x_d),(y_1,\dots,y_d)}|.$$

It has been shown that $||.||_a$ is a matrix norm which can compute the best advantage. Namely we have

$$\max_{\substack{A \text{ limited to } d \text{ queries} \\ \text{chosen plaintext, attack}}} \operatorname{Adv}^{A}(F,G) = \frac{1}{2} ||[F]^{d} - [G]^{d}||_{a}.$$
(1)

(See [24].)

Similarly, we recursively define the $||.||_s$ norm by

$$||A||_s = \max\left(\max_{x_1} \sum_{y_1} \pi_{x_1,y_1}(A), \max_{y_1} \sum_{x_1} \pi_{x_1,y_1}(A)\right)$$

(the norm of a matrix reduced to one entry being its absolute value) where $\pi_{x_1,y_1}(A)$ denotes the matrix in $\mathbf{R}^{\mathcal{M}_1^{d-1}\times\mathcal{M}_2^{d-1}}$ such that

$$(\pi_{x_1,y_1}(A))_{(x_2,\dots,x_d),(y_2,\dots,y_d)} = A_{(x_1,\dots,x_d),(y_1,\dots,y_d)}.$$

Then we have

$$\max_{\substack{A \text{ limited to } d \text{ queries} \\ \text{chosen plaintext and ciphertext attack}} \operatorname{Adv}^{\mathcal{A}}(F,G) = \frac{1}{2} ||[F]^d - [G]^d||_s.$$
 (2)

(See [24].)

1.3 Decorrelation Biases

We also use the decorrelation bias of order d of a function in the sense of a given norm ||.|| defined by

 $\operatorname{DecF}_{||.||}^{d}(F) = ||[F]^{d} - [F^{*}]^{d}||$

where F^* is a random function uniformly distributed, and the decorrelation bias of order d of a permutation defined by

$$\operatorname{DecP}_{||.||}^d(C) = ||[C]^d - [C^*]^d||$$

where C^* is a random permutation uniformly distributed. (See [18,20,23,24].)

2 On the Need for Orthomorphisms

Let us first consider the Λ^{σ} construction when σ is the identity function. Obviously if $(z,t) = \Lambda^{\sigma}(F_1,\ldots,F_r)(x,y)$ we have z-t=x-y. Thus, for any random round functions, $\Lambda^{\sigma}(F_1,\ldots,F_r)$ is fairly easily distinguishable with only one known plaintext. This is why we have to introduce the σ permutation.

Let us consider a one-round Lai-Massey scheme with σ :

$$(z,t) = (\sigma(x + F(x - y)), y + F(x - y)).$$

We have

$$z - t = (\sigma(x + F(x - y)) - (x + F(x - y))) + (x - y)$$

= $\sigma'(x + F(x - y)) + x - y$

where $\sigma'(u) = \sigma(u) - u$. Thus, if F is uniformly distributed and σ' is a permutation, then z - t is uniformly distributed. Ideally we thus require that σ and σ' are permutations, which means that σ is an orthomorphism of the group.

Unfortunately, the existence of orthomorphisms is not guaranteed for arbitrary groups. Actually, Hall-Paige Theorem [7] states that an Abelian finite group has an orthomorphism if and only if its order is odd or \mathbb{Z}_2^2 is isomorphic to one of its subgroups. In particular, \mathbb{Z}_{2^m} has no orthomorphism. In odd-ordered groups G, with multiplicative notations, the square $\sigma(x) = x^2$ is an orthomorphism since σ' is the identity permutation and σ is a permutation (its inverse is the $\frac{1+\#G}{2}$ -power function). In \mathbb{Z}_2^m with m>1, Schnorr and Vaudenay [15,16] exhibited

$$\sigma(x) = (x \text{ AND } c) \text{ XOR ROTL}^{i}(x)$$

which is an orthomorphism when the AND of all ROTL^{ij}(c) values is zero and the OR is 11...1. For instance, i = 1 and c = 00...01 leads to an orthomorphism. Stern and Vaudenay used a similar construction in CS-Cipher [17].

We thus relax the orthomorphism properties by adopting the following notion of α -almost orthomorphism.

¹ Throughout this paper OR, AND and XOR denote the usual bit-wise boolean operators on bitstrings of equal length, and ROTL^i denotes the left circular rotation by i positions.

Definition 1. In a given group G of order g, a permutation σ is called an α -almost orthomorphism if the function $\sigma'(x) = \sigma(x) - x$ is such that there are at most α elements in G with no preimage by σ' .

This definition fits to Patarin's notion of "spreading" [12,13]. We prefer here to emphasis on the approximation of orthomorphism properties.

We notice that since $(\sigma^{-1})'(x) = -\sigma'(\sigma^{-1}(x))$, then σ^{-1} is also an α -almost orthomorphism when σ is an α -almost orthomorphism.

Here is an useful lemma.

Lemma 2. If σ is an α -almost orthomorphism over the group G, then

$$\forall \delta \in G \setminus \{0\} \Pr_{(X,Y) \in {}_{U}G^{2}} [\sigma'(X) - \sigma'(Y) = \delta] \le \max(\alpha, 1)g^{-1}$$
 (3)

$$\forall \delta \in G \setminus \{0\} \ \Pr_{X \in_U G} [\sigma'(X) = \sigma'(X + \delta)] \le \alpha g^{-1}$$
 (4)

$$\forall \delta \in G \ \Pr_{X \in_{U} G} [\delta - \sigma'(X) / \mathfrak{G}'(G)] \le 2\alpha g^{-1}. \tag{5}$$

Proof. It is straightforward that for any set A, the number of preimages x such that $\sigma'(x) \in A$ is at most $\alpha + \#A$. Let n_y denote the number of preimages of y. We have

$$\Pr_{(X,Y)\in_U G^2} [\sigma'(X) - \sigma'(Y) = \delta] = g^{-2} \sum_u n_u n_{u+\delta}.$$

First, if $\alpha = 1$, for $\delta \neq 0$, the number of (x, y) pairs such that $\sigma'(x) - \sigma'(y) = \delta$ is at most g which is equal to αg .

Let us now consider $\alpha \geq 2$. If there exists one y such that $n_y = \alpha + 1$, then for all other ys we have $n_y \leq 1$. Hence

$$\Pr_{(X,Y)\in_U G^2}[\sigma'(X) - \sigma'(Y) = \delta] \le \frac{\alpha + 1}{g^2} - g^{-2} + g^{-2} \sum_u n_{u+\delta}$$
$$= \alpha g^{-2} + g^{-1}$$
$$\le \alpha g^{-1}.$$

In the other cases, we have $n_y \leq \alpha$ hence

$$\Pr_{(X,Y)\in_U G^2} [\sigma'(X) - \sigma'(Y) = \delta] \le g^{-2} \alpha \sum_u n_{u+\delta} = \alpha g^{-1}.$$

Therefore, in all cases this inequality holds.

We have

$$\Pr_{X \in UG}[\sigma'(X) = \sigma'(X + \delta)] \le \sum_{y; n_y \ge 2} n_y g^{-1} = 1 - g^{-1} \# \{y; n_y = 1\}.$$

The number of ys such that $n_y = 1$ is greater than $g - 2\alpha$, thus the probability is less than $2\alpha g^{-1}$.

The number of xs such that $\delta - \sigma'(x) / \mathfrak{G}'(G)$ is at most $\alpha + g - \#\sigma'(G)$ which is at most 2α .

As an example of almost orthomorphism in \mathbf{Z}_{2^m} (which has no orthomorphism), we claim that the simple rotation ROTL is a 1-almost orthomorphism. Actually, it is a permutation, and $\mathrm{ROTL}'(x)$ is equal to $x + \mathrm{MSB}(x)$ where $\mathrm{MSB}(x)$ denotes the most significant bit of x. The 0 value is taken twice by this function (by x=0 and $x=11\ldots 1$), the value 100...0 is never taken, and all the other values are taken once.

3 Extending the Luby-Rackoff Theorem

In order to extend Luby-Rackoff Theorem to the Lai-Massey scheme, we need the following lemma, which corresponds to Patarin's "coefficient H technique" [12,13].

Lemma 3. Let F_1^*, F_2^*, F_3^* be three independent random functions on a group G with uniform distribution, and let d be a positive integer. Let σ be an α -almost orthomorphism on G. For any family of G^2 elements $(x_1, \ldots, x_d, y_1, \ldots, y_d)$ such that the x_i values are pairwise different as well as the $y_i^l - y_i^r$ values, we have

$$\frac{\Pr[\Lambda^{\sigma}(F_1^*, F_2^*, F_3^*)(x_i) = y_i; i]}{\Pr[C^*(x_i) = y_i; i]} \ge 1 - \frac{d(d-1)}{2}(g^{-1} + g^{-2}) - f(\alpha)$$

where g denotes the cardinality of G and C^* is a random permutation of G^2 uniformly distributed, provided that $d < g^2$, and $f(\alpha)$ is a function such that f(0) = 0 and

$$f(\alpha) = d\frac{d(\alpha - 1) + 3\alpha - 1}{2a}$$
 for $\alpha > 0$.

Proof. We let U_i, V_i, W_i denote the values after the first, second and final round of $\Lambda^{\sigma}(F_1^*, F_2^*, F_3^*)(x_i)$ respectively. For any value t in G^2 , we let Δt denote $t^l - t^r$. The probabilistic event $[W_i = y_i]$ is equivalent to $[\Delta V_i = \Delta y_i]$ and $W_i^l = y_i^l$. Now we have

$$\Delta V_i = \sigma'(U_i^l + F_2^*(\Delta U_i)) + \Delta U_i$$

$$W_i^l = V_i^l + F_3^*(\Delta V_i).$$

The $[W_i = y_i]$ event is thus equivalent to

$$e_i = [F_2^*(\Delta U_i) \in {\sigma'}^{-1}(\Delta y_i - \Delta U_i) - V_i^l \text{ and } F_3^*(\Delta y_i) = y_i^l - U_i^l].$$

When the ΔU_i are pairwise different, as well as the ΔV_i , it is thus easy to compute the probability that we have $W_i = y_i$ for all i because it relies on independent $F_2(\Delta U_i)$ and $F_3(\Delta V_i)$ uniformly distributed random variables. In addition we need all $\Delta y_i - \Delta U_i$ to have preimages by σ' .

We have

$$Pr[W_i = y_i; i = 1, \dots, d]$$

=
$$Pr[e_i; i = 1, \dots, d]$$

$$\geq \Pr[e_i, \Delta U_i \neq \Delta U_j, \Delta y_i - \Delta U_i \in \sigma'(G); i \neq j]$$

$$= \Pr[e_i/\Delta U_i \neq \Delta U_j, \Delta y_i - \Delta U_i \in \sigma'(G); i \neq j] \times$$

$$\Pr[\Delta U_i \neq \Delta U_j, \Delta y_i - \Delta U_i \in \sigma'(G); i \neq j]$$

$$= g^{-2d}(1 - \Pr[\exists i < j \ \Delta U_i = \Delta U_j \ \text{or} \ \exists i \ \Delta y_i - \Delta U_i / \mathscr{E}'(G))])$$

which is greater than g^{-2d} times

$$1 - \frac{d(d-1)}{2} \cdot \max_{i < j} \Pr[\Delta U_i = \Delta U_j] - d \cdot \max_i \Pr[\Delta y_i - \Delta U_i / \mathfrak{C}'(G)].$$

We notice that

$$\Delta U_i = \sigma'(x_i^l + F(\Delta x_i)) + \Delta x_i.$$

The probability of having collisions with σ' with two different uniformly distributed inputs is less than $\max(\alpha,1)g^{-1}$ for $\Delta x_i \neq \Delta x_j$ from Equation (3). If we have $\Delta x_i = \Delta x_j$, then we will have $\Delta U_i = \Delta U_j$ with probability at most αg^{-1} from Equation (4) since $x_i \neq x_j$ and thus $x_i^l \neq x_j^l$. In addition, $\Pr[\Delta y_i - \Delta U_i] \neq \sigma'(G)$ is less than $2\alpha g^{-1}$ from Equation (5). Therefore $\Pr[W_i = y_i; i = 1, \ldots, d]$ is greater than

$$g^{-2d}\left(1-\frac{d(d-1)}{2}\max(\alpha,1)g^{-1}-2d\alpha g^{-1}\right).$$

We have

$$\Pr[C^*(x_i) = y_i; i = 1, \dots, d] = \frac{1}{g^2(g^2 - 1) \dots (g^2 - d + 1)}.$$

Since

$$\frac{g^2(g^2-1)\dots(g^2-d+1)}{g^{2d}} \ge 1 - \frac{d(d-1)}{2g^2}$$

when $g^2 > d$, we obtain the result.

We can now state our result.

Theorem 4. Let F_1^*, F_2^*, F_3^* be three independent random functions on a group G with a uniform distribution. Let σ be an α -almost orthomorphism on G. For any distinguisher limited to d chosen plaintexts $(d < g^2)$ between $\Lambda^{\sigma}(F_1^*, F_2^*, F_3^*)$ and a random permutation C^* with a uniform distribution, we have

$$Adv(\Lambda^{\sigma}(F_1^*, F_2^*, F_3^*), C^*) \le d(d-1)\left(g^{-1} + g^{-2}\right) + f(\alpha)$$

where g is the cardinality of G and $f(\alpha)$ is defined as in Lemma 3.

Proof. We can assume without loss of generality that the distinguisher never request the same query twice. Let ω denote the random tape of the distinguisher \mathcal{A} , and A be the set of all (ω, y) entries which leads to the output 1. We have

$$p^{\mathcal{O}} = \Pr\left[\mathcal{A}^{\mathcal{O}} = 1\right] = \sum_{(\omega, y) \in A} \Pr[\omega] \Pr[C(x_i) = y_i; i = 1, \dots, d]$$

where $x = (x_1, \ldots, x_d)$ in which x_i depends on ω and (y_1, \ldots, y_{i-1}) . We let $C = \Lambda^{\sigma}(F_1^*, F_2^*, F_3^*)$. Thus we have

$$p^{C} - p^{C^*} = \sum_{(\omega, y) \in A} \Pr[\omega](\Pr[C(x_i) = y_i; i] - \Pr[C^*(x_i) = y_i; i]).$$

We split the sum between the y entries for which the Δy_i are pairwise different, and the others. From the previous lemma we have

$$p^C - p^{C^*} \ge -\sum_{\substack{(\omega, y) \in A \\ \Delta y_i \ne \Delta y_j}} \Pr[\omega] p^* \epsilon - \Pr[\exists i < j \ \Delta C^*(y_i) = \Delta C^*(y_j)]$$

where $\epsilon = \frac{d(d-1)}{2}(g^{-1} + g^{-2}) + f(\alpha)$ and p^* is the probability that $C^*(x_i) = y_i$ for i = 1, ..., d. The first sum is less than ϵ , and the last probability is less than $\frac{d(d-1)}{2}g^{-1}$, thus

$$p^{C} - p^{C^*} \ge -\epsilon - \frac{d(d-1)}{2}g^{-1}.$$

We can then apply the same result to the symmetric distinguisher, and obtain the result. $\hfill\Box$

4 Inheritance of Decorrelation in the Lai-Massey Scheme

We can use the same proof as in [24] for proving that the decorrelation bias of the round functions of a Lai-Massey scheme is inherited by the whole structure. The following lemma is a straightforward application of a more general lemma from [24].

Lemma 5. Let m be an integer, and F_1, \ldots, F_r be r independent random functions on a group G. Let σ be a permutation on G. We have

$$||[\Lambda^{\sigma}(F_1,\ldots,F_r)]^d - [\Lambda^{\sigma}(F_1^*,\ldots,F_r^*)]^d||_a \le \sum_{i=1}^r \operatorname{DecF}_{||.||_a}^d(F_i)$$

where F_1^*, \ldots, F_r^* are uniformly distributed random functions.

Following [24], this lemma and Lemma 3 enables to prove the following corollary.

Corollary 6. If F_1, \ldots, F_r are r (with $r \geq 3$) independent random functions on a group G of order g such that $\mathrm{DecF}^d_{||.||_a}(F_i) \leq \epsilon$ and if σ is an α -almost orthomorphism on G, we have

$$\operatorname{DecP}_{||.||_a}^d(\Lambda^{\sigma}(F_1,\ldots,F_r)) \leq \left(3\epsilon + d(d-1)\left(2g^{-1} + g^{-2}\right) + 2f(\alpha)\right)^{\left\lfloor \frac{r}{3}\right\rfloor}$$

where $f(\alpha)$ is defined in Lemma 3.

5 On Super-Pseudorandomness

Super-pseudorandomness corresponds to cases where attacks can query chosen ciphertexts as well. We extend Lemma 3 in order to get results on the super-pseudorandomness.

Lemma 7. Let $F_1^*, F_2^*, F_3^*, F_4^*$ be four independent random functions on a group G with uniform distribution, and let d be an integer. Let σ be an α -almost orthomorphism on G. For any set of $x_1, \ldots, x_d, y_1, \ldots, y_d$ values in G^2 such that the x_i values are pairwise different, we have

$$\frac{\Pr[\Lambda^{\sigma}(F_1^*, F_2^*, F_3^*, F_4^*)(x_i) = y_i; i]}{\Pr[C^*(x_i) = y_i; i]} \ge 1 - d(d-1)\left(g^{-1} + g^{-2}\right) - f'(\alpha)$$

where g denotes the cardinality of G and C* is a random permutation of G^2 uniformly distributed, provided that $d < g^2$, and $f'(\alpha)$ is a function such that f'(0) = 0 and

$$f'(\alpha) = dg^{-1}(d(\alpha - 1) + \alpha - 1)$$
 for $\alpha > 0$.

Proof. $\Lambda^{\sigma}(F_1^*, F_2^*, F_3^*, F_4^*)(x_i) = y_i)$ is equivalent to

$$\Lambda^{\sigma}(F_1^*, F_2^*, F_3^*)(x_i) = \Lambda^{\sigma^{-1}}(F_4^*)(y_i).$$

We can focus on the probability that all $\Delta \Lambda^{\sigma^{-1}}(F_4^*)(y_i)$ are pairwise different. Similarly as in the proof of Lemma 3, this holds but for a probability less than $\frac{d(d-1)}{2} \max(\alpha, 1)g^{-1}$. We can then apply Lemma 3 to complete the proof.

This extends Theorem 4.

Theorem 8. Let $F_1^*, F_2^*, F_3^*, F_4^*$ be four independent random functions on a group G with a uniform distribution. Let σ be an α -almost orthomorphism on G. For any distinguisher limited to d chosen plaintexts or ciphertexts ($d < g^2$) between $\Lambda^{\sigma}(F_1^*, F_2^*, F_3^*, F_4^*)$ and a random permutation C^* with a uniform distribution, we have

$$Adv(\Lambda^{\sigma}(F_1^*, F_2^*, F_3^*, F_4^*), C^*) \le d(d-1)\left(g^{-1} + g^{-2}\right) + f'(\alpha)$$

where g denotes the cardinality of G and $f'(\alpha)$ is defined in Lemma 7.

The proof is the same as in Theorem 4, but with no consideration on the $\Delta y_i \neq \Delta y_j$ cases.

This shows that a 4-round random Lai-Massey scheme with an α -almost orthomorphism is a super-pseudorandom permutation when it is used less than $\sqrt{g/\max(\alpha,1)}$ times. This also extends to the following decorrelation bias upper bound.

Corollary 9. If F_1, \ldots, F_r are r (with $r \geq 4$) independent random functions on a group G of order g such that $\operatorname{DecF}^d_{||\cdot||_a}(F_i) \leq \epsilon$ and if σ is an α -almost orthomorphism on G, we have

$$\operatorname{DecP}_{||.||_{s}}^{d}(\Lambda^{\sigma}(F_{1},\ldots,F_{r})) \leq \left(4\epsilon + d(d-1)\left(2g^{-1} + g^{-2}\right) + 2f'(\alpha)\right)^{\left\lfloor \frac{r}{4} \right\rfloor}$$

where $f'(\alpha)$ is defined in Lemma 7.

6 A New Family of Block Ciphers

In this section we construct a new family of block ciphers called Walnut (as for "Wonderful Algorithm with Light N-Universal Transformation") Walnut is a Lai-Massey scheme which depends on four parameters (m, r, d, q) where m is the message-block length (must be even), r is the number of rounds, d is the order of decorrelation and q is an integral prime power at least $2^{\frac{m}{2}}$. It is characterized by having round function F_i with the form

$$F_i(x) = \pi_i(r_i(K_{i,1}) + r_i(K_{i,2})r_i(x) + \ldots + r_i(K_{i,d})r_i(x)^{d-1})$$

where the $K_{i,j}$ are independent uniformly distributed bitstrings of length m/2, r_i is an injective mapping from $\{0,1\}^{\frac{m}{2}}$ to $\mathrm{GF}(q)$, and π_i is a surjective mapping from $\mathrm{GF}(q)$ to $\{0,1\}^{\frac{m}{2}}$. This is a straightforward extension of the Peanut construction. It has been shown in [24] that $\mathrm{DecF}^d(F_i)$ is less than

$$\epsilon = 2\left((1+\delta)^d - 1\right)$$

where $q = (1+\delta)2^{\frac{m}{2}}$. We use $\sigma = \text{ROTL}$ as a 1-almost orthomorphism. Therefore by approximating the upper bounds of Corollaries 6 and 9 we have

$$\begin{aligned} &\operatorname{DecP}^d_{||.||_a}(\operatorname{Walnut}(m,r,d,q)) \leq \sim \left(6d\delta + 2d^2 2^{-\frac{m}{2}}\right)^{\left\lfloor \frac{r}{3} \right\rfloor} \\ &\operatorname{DecP}^d_{||.||_s}(\operatorname{Walnut}(m,r,d,q)) \leq \sim \left(8d\delta + 2d^2 2^{-\frac{m}{2}}\right)^{\left\lfloor \frac{r}{4} \right\rfloor}. \end{aligned}$$

With m = 64, d = 2 and $p = 2^{32} + 15$, we obtain

$$\operatorname{DecP}_{||.||_a}^d(\operatorname{Walnut}(64, r, 2, 2^{32} + 15)) \le 2^{-24 \left\lfloor \frac{r}{3} \right\rfloor}$$

 $\operatorname{DecP}_{||.||_a}^d(\operatorname{Walnut}(64, r, 2, 2^{32} + 15)) \le 2^{-24 \left\lfloor \frac{r}{4} \right\rfloor}.$

This provides sufficient security against differential and linear attacks for $r \geq 12$.

7 Conclusion

We have shown that adding a simple orthomorphism (or almost orthomorphism) enables the Lai-Massey scheme to provide randomness on three rounds, and super-pseudorandomness on four rounds, like for the Feistel scheme. We have shown that we can get similar decorrelation upper bounds as well and propose a new block cipher family.

Acknowledgement

I wish to thank NTT and Tatsuaki Okamoto for providing a good environment for research activities, and his team for enjoyable meetings and fruitful discussions.

References

- FIPS 46, Data Encryption Standard. U.S. Department of Commerce National Bureau of Standards, National Technical Information Service, Springfield, Virginia. Federal Information Processing Standard Publication 46, 1977.
- O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, R. Harley, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. DFC Update. In Proceedings from the Second Advanced Encryption Standard Candidate Conference, National Institute of Standards and Technology (NIST), March 1999.
- J. Daemen, L. Knudsen, V. Rijmen. The Block Cipher Square. In Fast Software Encryption, Haifa, Israel, Lecture Notes in Computer Science 1267, pp. 149–171, Springer-Verlag, 1997.
- H. Feistel. Cryptography and Computer Privacy. Scientific American, vol. 228, pp. 15–23, 1973.
- H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern,
 S. Vaudenay. Decorrelated Fast Cipher: an AES Candidate. (Extended Abstract.)
 In Proceedings from the First Advanced Encryption Standard Candidate Conference, National Institute of Standards and Technology (NIST), August 1998.
- H. Gilbert, M. Girault, P. Hoogvorst, F. Noilhan, T. Pornin, G. Poupard, J. Stern, S. Vaudenay. Decorrelated Fast Cipher: an AES Candidate. Submitted to the Advanced Encryption Standard process. In CD-ROM "AES CD-1: Documentation", National Institute of Standards and Technology (NIST), August 1998.
- M. Hall, L. J. Paige. Complete Mappings of Finite Groups. In Pacific Journal of Mathematics, vol. 5, pp. 541–549, 1955.
- 8. X. Lai. On the Design and Security of Block Ciphers, ETH Series in Information Processing, vol. 1, Hartung-Gorre Verlag Konstanz, 1992.
- X. Lai, J. L. Massey. A Proposal for a New Block Encryption Standard. In Advances in Cryptology EUROCRYPT '90, Aarhus, Denemark, Lecture Notes in Computer Science 473, pp. 389–404, Springer-Verlag, 1991.
- M. Luby, C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM Journal on Computing, vol. 17, pp. 373–386, 1988.
- J. L. Massey. SAFER K-64: a Byte-Oriented Block-Ciphering Algorithm. In Fast Software Encryption, Haifa, Israel, Lecture Notes in Computer Science, 1267, pp. 1–17, Springer-Verlag, 1994.
- J. Patarin. Etude des Générateurs de Permutations Basés sur le Schéma du D.E.S.,
 Thèse de Doctorat de l'Université de Paris 6, 1991. 12, 13
- J. Patarin. How to Construct Pseudorandom and Super Pseudorandom Permutations from One Single Pseudorandom Function. In Advances in Cryptology EURO-CRYPT '92, Balatonfüred, Hungary, Lecture Notes in Computer Science 658, pp. 256–266, Springer-Verlag, 1993. 12, 13
- B. Schneier. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). In Fast Software Encryption, Cambridge, United Kingdom, Lecture Notes in Computer Science 809, pp. 191–204, Springer-Verlag, 1994.
- C. P. Schnorr, S. Vaudenay. Parallel FFT-Hashing. In Fast Software Encryption, Cambridge, United Kingdom, Lecture Notes in Computer Science 809, pp. 149–156, Springer-Verlag, 1994. 11
- C. P. Schnorr, S. Vaudenay. Black Box Cryptanalysis of Hash Networks based on Multipermutations. In *Advances in Cryptology EUROCRYPT '94*, Perugia, Italy, Lecture Notes in Computer Science 950, pp. 47–57, Springer-Verlag, 1995. 11

- J. Stern, S. Vaudenay. CS-Cipher. In Fast Software Encryption, Paris, France, Lecture Notes in Computer Science, 1372, pp. 189–205, Springer-Verlag, 1998.
- S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. In STACS 98, Paris, France, Lecture Notes in Computer Science 1373, pp. 249–275, Springer-Verlag, 1998.
 9, 11
- S. Vaudenay. Provable Security for Block Ciphers by Decorrelation. (Full Paper.)
 Technical report LIENS-98-8, Ecole Normale Supérieure, 1998.
 URL:ftp://ftp.ens.fr/pub/reports/liens/liens-98-8.A4.ps.Z
- 20. S. Vaudenay. Feistel Ciphers with L_2 -Decorrelation. sac, pp. 1–14, Springer-Verlag, 1998. 9, 11
- 21. S. Vaudenay. The Decorrelation Technique Home-Page. URL:http://www.dmi.ens.fr/~vaudenay/decorrelation.html 9
- S. Vaudenay. Vers une Théorie du Chiffrement Symétrique, Dissertation for the diploma of "habilitation to supervise research" from the University of Paris 7, Technical Report LIENS-98-15 of the Laboratoire d'Informatique de l'Ecole Normale Supérieure, 1998.
- S. Vaudenay. Resistance Against General Iterated Attacks. In Advances in Cryptology EUROCRYPT '99, Prague, Czech Republic, Lecture Notes in Computer Science 1592, pp. 255–271, Springer-Verlag, 1999.
- S. Vaudenay. Adaptive-Attack Norm for Decorrelation and Super-Pseudorandomness. Technical report LIENS-99-2, Ecole Normale Supérieure, 1999. (To appear in SAC' 99, LNCS, Springer-Verlag.)
 URL:ftp://ftp.ens.fr/pub/reports/liens/liens-99-2.A4.ps.Z 10, 11, 15, 17

On Cryptographically Secure Vectorial Boolean Functions

Takashi Satoh¹, Tetsu Iwata², and Kaoru Kurosawa²

Faculty of International Environmental Engineering
 Promotion and Development Office,
 Kitakyushu University
 4-2-1 Kitagata, Kokuraminami-ku, Kitakyushu 802-8577, Japan
 tsatoh@kitakyu-u.ac.jp
 Department of Electrical and Electronic Engineering,

Faculty of Engineering,

Tokyo Institute of Technology

2-12-1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan {tez,kurosawa}@ss.titech.ac.jp

Abstract. In this paper, we show the first method to construct vectorial bent functions which satisfy both the largest degree and the largest number of output bits simultaneously. We next apply this method to construct balanced vectorial Boolean functions which have larger non-linearities than previously known constructions.

1 Introduction

Boolean functions play an important role in block ciphers (for example, see [3,7,8,10,11]) and stream ciphers [12,2]. The nonlinearity N_f of a Boolean function $f(x_1,\ldots,x_n)$ is defined as a distance between f and the set of affine functions $\{a_0 \oplus a_1x_1 \oplus \cdots \oplus a_nx_n\}$. N_f should be large to resist the linear attack [2,6].

 $f(x_1, \ldots, x_n)$ is said to be a bent function if it has the maximum nonlinearity [5,9]. More generally, we say that a vectorial Boolean function $F(x_1, \ldots, x_n) = (f_1, \ldots, f_m)$ is a (n, m)-bent function if any nonzero linear combination of f_1, \ldots, f_m is a bent function. For (n, m)-bent functions, it is known that [7]

$$m \le n/2 \ . \tag{1}$$

On the other hand, the degree of f, deg(f), is defined as the degree of the highest degree term of the algebraic normal form:

$$f(x_1,\ldots,x_n) = a_0 \oplus \bigoplus_{1 \le i \le n} a_i x_i \oplus \bigoplus_{1 \le i < j \le n} a_{i,j} x_i x_j \oplus \cdots \oplus a_{1,2,\ldots,n} x_1 x_2 \cdots x_n.$$

The degree of a vectorial Boolean function $F(x_1, \ldots, x_n) = (f_1, \ldots, f_m)$ is defined as

$$\deg(F) \stackrel{\triangle}{=} \min_{\substack{(c_1,\ldots,c_m)\neq(0,\ldots,0)}} \deg(c_1 f_1 \oplus \cdots \oplus c_m f_m) .$$

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 20–28, 1999.
 © Springer-Verlag Berlin Heidelberg 1999

In block ciphers, deg(F) should be large to resist the higher order differential attack [4]. For (n, m)-bent functions, it is known that [5,9]

$$\deg(F) \le n/2 \ . \tag{2}$$

However, no construction method has been known so far which achieves both equalities of eq.(1) and eq.(2) simultaneously. In this paper, we show the first method to construct (n, m)-bent functions which satisfy the both equalities of eq.(1) and eq.(2) simultaneously.

It is known that bent functions are not balanced. For m=1, Seberry, Zhang and Zheng [10] and Dobbertin [3] showed balanced functions which have large nonlinearity. For m=n, Nyberg showed balanced *vectorial* Boolean functions with high nonlinearity [8].

We next apply our method to construct balanced vectorial Boolean functions with high nonlinearity. For $2 \le m \le n/2$, our balanced vectorial Boolean functions have larger nonlinearity than that of [8].

2 Bent Functions

For a Boolean function $f(x_1, \ldots, x_n)$, define

$$||f(x_1,\ldots,x_n)|| \stackrel{\triangle}{=} |\{(x_1,\ldots,x_n) \mid f(x_1,\ldots,x_n) = 1\}| .$$

$$N_f \stackrel{\triangle}{=} \min_{a_0,\ldots,a_n} ||f(x_1,\ldots,x_n) \oplus (a_0 \oplus a_1x_1 \oplus \cdots \oplus a_nx_n)|| .$$

 N_f is called the nonlinearity of f and it denotes a distance between f and the set of affine functions $\{a_0 \oplus a_1 x_1 \oplus \cdots \oplus a_n x_n\}$. For a vectorial Boolean function $F(x_1, \ldots, x_n) = (f_1, \ldots, f_m)$, the nonlinearity N_F is defined as

$$N_F \stackrel{\triangle}{=} \min_{(c_1, \dots, c_m) \neq (0, \dots, 0)} N_{c_1 f_1 \oplus \dots \oplus c_m f_m} . \tag{3}$$

 N_F should be large to resist the linear attack [2,6]. It is known that

$$N_f \le 2^{n-1} - 2^{\frac{n}{2}-1}$$
 and $N_F \le 2^{n-1} - 2^{\frac{n}{2}-1}$. (4)

Definition 2.1. $f(x_1,...,x_n)$ is called a bent function if $N_f = 2^{n-1} - 2^{\frac{n}{2}-1}$. $F(x_1,...,x_n) = (f_1,...,f_m)$ is called a (n,m)-bent function if $N_F = 2^{n-1} - 2^{\frac{n}{2}-1}$.

Proposition 2.1. [5,9] If $f(x_1,...,x_n)$ is a bent function, then n is even and

$$\deg(f) \le n/2 .$$

Proposition 2.2. [7] If $F(x_1, ..., x_n) = (f_1, ..., f_m)$ is a (n, m)-bent function, then n is even,

$$m \le n/2$$
 and $\deg(F) \le n/2$.

For even n, let $\mathcal{X} \stackrel{\triangle}{=} (x_1, \dots, x_{n/2})$ and $\mathcal{Y} \stackrel{\triangle}{=} (y_1, \dots, y_{n/2})$. Then it is known that

$$f(\mathcal{Y}, \mathcal{X}) = \pi(\mathcal{Y}) \cdot \mathcal{X}^T \oplus g(\mathcal{Y})$$

is a bent function if π is a permutation on $\{0,1\}^{n/2}$, where $g(\mathcal{Y})$ is any Boolean function [1]. This is called a Maiorana-McFarland type bent function [1]. From the definition of (n,m)-bent functions, we have the following proposition immediately.

Proposition 2.3. [7] $F(\mathcal{Y}, \mathcal{X}) = (f_1, \dots, f_m)$ is a (n, m)-bent function if

$$f_i(\mathcal{Y}, \mathcal{X}) = \pi_i(\mathcal{Y}) \cdot \mathcal{X}^T \oplus g_i(\mathcal{Y})$$

and every nonzero linear combination of $\{\pi_i\}$ is a permutation on $\{0,1\}^{n/2}$, where $g_i(\mathcal{Y})$ is any Boolean function.

Nyberg gave several constructions of such $\{\pi_i\}$ [7].

3 Proposed Vectorial Bent Function

3.1 Notation

For a binary vector (y_1, \ldots, y_m) , define

$$dec(y_1, \dots, y_m) \stackrel{\triangle}{=} 2^{m-1}y_1 + 2^{m-2}y_2 + \dots + y_m$$
.

For an element α of $GF(2^m)$, let $[\alpha]$ denote a vector representation of α .

3.2 Proposed Construction

We now present a method to construct (n, m)-bent functions which satisfy both equalities of Proposition 2.2.

Proposition 3.1. [5, page 372] Any Boolean function f can be expanded as

$$f(x_1, \dots, x_n) = \bigoplus_{a_1, \dots, a_n} h(a_1, \dots, a_n) x_1^{a_1} \cdots x_n^{a_n}$$

where

$$h(a_1,\ldots,a_n) = \bigoplus_{b \subset a} f(b_1,\ldots,b_n) ,$$

and $b \subset a$ means that the 1's in (b_1, \ldots, b_n) are a subset of the 1's in (a_1, \ldots, a_n) .

Lemma 3.1. Let α be a primitive element of $GF(2^m)$. Then

$$1 + \alpha + \alpha^2 + \dots + \alpha^l \begin{cases} \neq 0 & \text{if } 0 < l + 1 < 2^m - 1 \\ = 0 & \text{if } l + 1 = 2^m - 1 \end{cases}$$

Proof. Since α is a primitive element, we have

$$(1+\alpha)(1+\alpha+\alpha^2+\cdots+\alpha^l) = 1+\alpha^{l+1} \begin{cases} \neq 0 & \text{if } 0 < l+1 < 2^m-1 \\ = 0 & \text{if } l+1 = 2^m-1 \end{cases},$$

Therefore, this lemma holds.

For even n, let m = n/2, $\mathcal{X} = (x_1, \dots, x_m)$ and $\mathcal{Y} = (y_1, \dots, y_m)$. Let α be a primitive element of $GF(2^{n/2})$. Consider $F(\mathcal{Y}, \mathcal{X}) = (f_1, \dots, f_{n/2})$ such that

$$f_i(\mathcal{Y}, \mathcal{X}) = [\varphi_i(\mathcal{Y})] \cdot \mathcal{X}^T \oplus g_i(\mathcal{Y}) ,$$

where

$$\varphi_i(\mathcal{Y}) \stackrel{\triangle}{=} \begin{cases} 0 & \text{if } \mathcal{Y} = (0, \dots, 0) \\ \alpha^{dec(\mathcal{Y})+i-1} & \text{otherwise} \end{cases}$$

and g_i is any Boolean function.

Theorem 3.1. The above F is a (n,m)-bent function such that m=n/2 and deg(F)=n/2.

Proof. For any $c = (c_1, ..., c_m) \neq (0, ..., 0)$, let

$$\Phi_c(\mathcal{Y}) \stackrel{\triangle}{=} c_1 \varphi_1(\mathcal{Y}) + \dots + c_m \varphi_m(\mathcal{Y}) . \tag{5}$$

Then it is easy to see that

$$\Phi_c(\mathcal{Y}) = \begin{cases}
0 & \text{if } \mathcal{Y} = (0, \dots, 0) \\
\alpha^{dec(\mathcal{Y})} \gamma & \text{otherwise}
\end{cases},$$
(6)

where

$$\gamma \stackrel{\triangle}{=} (c_1 + c_2 \alpha + \dots + c_m \alpha^{m-1}) \neq 0 \tag{7}$$

because α is a primitive element of $GF(2^m)$. This implies that $[\Phi_c(\mathcal{Y})]$ is a permutation on $\{0,1\}^m$. Therefore, F is a (n,n/2)-bent function from Proposition 2.3.

Next suppose that $[\Phi_c(\mathcal{Y})]$ is written as

$$[\Phi_c(\mathcal{Y})] = h(1, \dots, 1)y_1 \cdots y_m$$

$$\oplus h(1, \dots, 1, 0)y_1 \cdots y_{m-1}$$

$$\oplus \cdots \oplus h(0, 1, \dots, 1)y_2 \cdots y_m$$

$$\oplus \cdots \oplus h(0, \dots, 0) .$$

Let $\beta \stackrel{\triangle}{=} 1 + \alpha + \alpha^2 + \dots + \alpha^{2^{m-1}-1}$. Then

$$\beta = 1 + \alpha + \alpha^2 + \dots + \alpha^{2^{m-1}-1} = \sum_{(i_2,\dots,i_m)} \alpha^{dec(0,i_2,\dots,i_m)}$$
.

From eq.(6)

$$\gamma \beta = \gamma \left(\sum_{(i_2, \dots, i_m)} \alpha^{dec(0, i_2, \dots, i_m)} \right) = \sum_{(i_2, \dots, i_m)} \Phi_c(0, i_2, \dots, i_m) .$$

Finally, from Proposition 3.1, we have

$$h(0,1,\ldots,1) = \bigoplus_{(i_2,\ldots,i_m)} [\Phi_c(0,i_2,\ldots,i_m)] = [\gamma\beta] .$$
 (8)

Hence, we have

$$[\Phi_c(\mathcal{Y})] \cdot \mathcal{X}^T = [\gamma \beta] \cdot \mathcal{X}^T y_2 \cdots y_m \oplus \cdots , \qquad (9)$$

where $\gamma\beta \neq 0$ from Lemma 3.1 and eq.(7). Therefore,

$$\deg([\Phi_c(\mathcal{Y})] \cdot \mathcal{X}^T) \ge \deg([\gamma \beta] \cdot \mathcal{X}^T y_2 \cdots y_m) = m = n/2.$$

This means that $\deg(F) = n/2$ since $\deg(F) \le n/2$ from Proposition 2.2.

3.3 Maximum Degree for Each Variable

Definition 3.1. We say that a (n,m)-bent function $F(x_1,\ldots,x_n)=(f_1,\ldots,f_m)$ has the maximum degree for each variable if each variable x_i appears in some term of degree n/2.

Definition 3.2. [5, page 120] A normal basis of $GF(p^k)$ is a basis of the form $\beta, \beta^p, \dots, \beta^{p^k-1}$.

Proposition 3.2. [5, page 122] A normal basis exists in any field $GF(p^k)$.

Theorem 3.2. In the proposed construction, let m = n/2 and let

$$\beta = 1 + \alpha + \alpha^2 + \dots + \alpha^{2^{m-1} - 1} . \tag{10}$$

Then our (n,m)-bent function F has the maximum degree for each variable if $\{\beta, \beta^2, \dots, \beta^{2^{m-1}}\}$ is a normal basis of $GF(2^m)$.

Proof. In eq.(8), we have proved that

$$h(0,1,\ldots,1) = [\gamma\beta] .$$

Similarly, we can prove that

$$h(1,...,1,0) = [\gamma \beta^2],$$

 $h(1,...,1,0,1) = [\gamma \beta^{2^2}],$
:

Then eq.(9) becomes as follows.

$$[\Phi_c(\mathcal{Y})] \cdot \mathcal{X}^T = [\gamma \beta^2] \cdot \mathcal{X}^T y_1 \cdots y_{m-1} \oplus \cdots \oplus [\gamma \beta] \cdot \mathcal{X}^T y_2 \cdots y_m \oplus \cdots$$

Now $[\gamma\beta^2], [\gamma\beta^{2^2}], \dots, [\gamma\beta]$ are linearly independent since $\{\beta, \beta^2, \dots, \beta^{2^{m-1}}\}$ is a normal basis and $\gamma \neq 0$. This means that each x_i is included in some term of degree m = n/2. It is clear that each y_i is included in some term of degree n/2.

Corollary 3.1. Our (n, n/2)-bent function F has the maximum degree for each variable if $2^{n/2} - 1$ is a prime.

Proof. There exists a normal basis $\beta, \beta^2, \beta^{2^2}, \dots, \beta^{2^{n/2-1}}$ in $GF(2^{n/2})$ from Proposition 3.2. On the other hand, if eq.(10) holds, then from lemma 3.1,

$$(1 + \alpha^{2^{m-1}})\beta = 1 + \alpha + \alpha^2 + \dots + \alpha^{2^m - 1} = \alpha^{2^m - 1} = 1$$

and

$$(1 + \alpha^{2^m})\beta^2 = 1$$
$$(1 + \alpha)\beta^2 = 1$$
$$\alpha = \beta^{-2} + 1.$$

Now any nonzero element is a primitive element of $GF(2^{n/2})$ if $2^{n/2} - 1$ is a prime. Therefore, $\alpha = \beta^{-2} + 1$ is a primitive element. This implies that the condition of Theorem 3.2 is satisfied.

4 Application to Balanced Boolean Functions

We say that $f(x_1, \ldots, x_n)$ is balanced if

$$||f(x_1,\ldots,x_n)|| = 2^{n-1}$$
.

We also say that $F(x_1, \ldots, x_n) = (f_1, \ldots, f_m)$ is balanced if any nonzero linear combination of f_1, \ldots, f_m is balanced.

For m=1, Seberry, Zhang and Zheng [10] and Dobbertin [3] showed balanced functions which have large nonlinearity. For m=n, Nyberg showed balanced vectorial Boolean functions with high nonlinearity such as follows.

Proposition 4.1. [8] For m = n, there exists a balanced vectorial Boolean function such that

$$N_F \left\{ \begin{split} & \geq 2^{n-1} - 2^{\frac{n}{2}} & \text{if n is even }, \\ & = 2^{n-1} - 2^{\frac{n-1}{2}} & \text{if n is odd }. \end{split} \right.$$

This section shows that we can obtain balanced *vectorial* Boolean functions which have larger nonlinearity than Proposition 4.1 for $2 \le m \le n/2$ by applying our technique of Sec.3.2.

Theorem 4.1. Suppose that there exists a balanced vectorial Boolean function $F(x_1, \ldots, x_h) = (f_1, \ldots, f_m)$ with nonlinearity N_F for $m \leq h$. Then there exists a balanced vectorial Boolean function $\widetilde{F} = (\widetilde{f}_1, \ldots, \widetilde{f}_m)$ with 2h input variables such that

$$N_{\widetilde{F}} \ge N_F + 2^{h-1}(2^h - 2)$$
.

Proof. Let $\mathcal{X} = (x_1, \dots, x_h)$ and $\mathcal{Y} = (y_1, \dots, y_h)$. Let α be a primitive element of $GF(2^h)$. Define

$$\widetilde{f}_i(\mathcal{Y}, \mathcal{X}) \stackrel{\triangle}{=} \begin{cases} f_i(\mathcal{X}) & \text{if } \mathcal{Y} = (0, \dots, 0) \\ \left[\alpha^{dec(\mathcal{Y}) + i - 1}\right] \cdot \mathcal{X}^T \oplus g_i(\mathcal{Y}) & \text{otherwise} \end{cases}.$$

where $g_i(\mathcal{Y})$ is any Boolean function. Let $\widetilde{F}(\mathcal{Y}, \mathcal{X}) \stackrel{\triangle}{=} (\widetilde{f}_1, \dots, \widetilde{f}_m)$. For any $\mathbf{c} = (c_1, \dots, c_m) \neq (0, \dots, 0)$, let

$$\widetilde{f}_{\mathbf{c}}(\mathcal{X}, \mathcal{Y}) \stackrel{\triangle}{=} c_{1} f_{1}(\mathcal{X}, \mathcal{Y}) \oplus \cdots \oplus c_{m} f_{m}(\mathcal{X}, \mathcal{Y})
= \begin{cases}
c_{1} f_{1}(\mathcal{X}) \oplus \cdots \oplus c_{m} f_{m}(\mathcal{X}) & \text{if } \mathcal{Y} = (0, \dots, 0) \\
\left(c_{1} [\varphi_{1}(\mathcal{Y})] \oplus \cdots \oplus c_{m} [\varphi_{m}(\mathcal{Y})]\right) \cdot \mathcal{X}^{T} \\
\oplus c_{1} g_{1}(\mathcal{Y}) \oplus \cdots c_{m} g_{i}(\mathcal{Y})
\end{cases} \text{ otherwise },$$

where $\varphi_i(\mathcal{Y}) = \alpha^{dec(\mathcal{Y})+i-1}$.

We first prove that $\widetilde{f}_{\mathbf{c}}(\mathcal{Y}, \mathcal{X})$ is balanced. For $\mathcal{Y} = (0, \dots, 0)$, $\widetilde{f}_{\mathbf{c}}(\mathcal{X}, 0, \dots, 0) = c_1 f_1 \oplus \dots \oplus c_m f_m$ is balanced since F is balanced. For $\mathcal{Y} \neq (0, \dots, 0)$,

$$c_1[\varphi_1(\mathcal{Y})] \oplus \cdots \oplus c_m[\varphi_m(\mathcal{Y})] = [\alpha^{dec(\mathcal{Y})}(c_1 + c_2\alpha + \cdots + c_m\alpha^{m-1})]$$
$$= [\alpha^{dec(\mathcal{Y})}\gamma] \neq (0, \dots, 0)$$
(11)

where $\gamma = c_1 + c_2 \alpha + \dots + c_m \alpha^{m-1}$. Note that $\gamma \neq 0$ since α is a primitive element of $GF(2^h)$. Therefore $(c_1[\varphi_1(\mathcal{Y})] \oplus \dots \oplus c_m[\varphi_m(\mathcal{Y})]) \cdot \mathcal{X}^T = [\alpha^{dec(\mathcal{Y})}\gamma] \cdot \mathcal{X}^T$ is balanced for each fixed $\mathcal{Y} \neq (0, \dots, 0)$. This implies that $\tilde{f}_{\mathbf{c}}(\mathcal{Y}, \mathcal{X})$ is balanced.

We next compute the nonlinearity of $f_{\mathbf{c}}(\mathcal{Y}, \mathcal{X})$. Let

$$L(\mathcal{Y}, \mathcal{X}) = \mathbf{a} \cdot \mathcal{Y}^T \oplus \mathbf{b} \cdot \mathcal{X}^T \oplus c_0 .$$

Then

$$\begin{split} N_{\widetilde{F}} &= \min_{L} ||\widetilde{f}_{\mathbf{c}}(\mathcal{Y}, \mathcal{X}) \oplus L(\mathcal{Y}, \mathcal{X})|| \\ &\geq \min_{L} ||\widetilde{f}_{\mathbf{c}}(0, \dots, 0, \mathcal{X}) \oplus L(0, \dots, 0, \mathcal{X})|| \\ &+ \min_{L} \sum_{\mathcal{Y} \neq (0, \dots, 0)} ||\widetilde{f}_{c}(\mathcal{Y}, \mathcal{X}) \oplus L(\mathcal{Y}, \mathcal{X})|| \\ &= \min_{\mathbf{b}, c_{0}} ||c_{1}f_{1}(\mathcal{X}) \oplus \dots \oplus c_{m}f_{m}(\mathcal{X}) \oplus \mathbf{b} \cdot \mathcal{X}^{T} \oplus c_{0}|| \\ &+ \min_{\mathbf{b}, c_{0}} \sum_{\mathcal{Y} \neq (0, \dots, 0)} ||(c_{1}[\varphi_{1}(\mathcal{Y})] \oplus \dots \oplus c_{m}[\varphi_{m}(\mathcal{Y})] \oplus \mathbf{b}) \cdot \mathcal{X}^{T} \oplus \widetilde{c}_{\mathcal{Y}}|| \\ &\geq N_{F} + \min_{\mathbf{b}, c_{0}} \sum_{\mathcal{Y} \neq (0, \dots, 0)} ||([\alpha^{dec(\mathcal{Y})} \gamma] \oplus \mathbf{b}) \cdot \mathcal{X}^{T} \oplus \widetilde{c}_{\mathcal{Y}}|| \end{split}$$

for some $\widetilde{c}_{\mathcal{Y}}$ (= 0 or 1) from eq.(11). For any **b**, there exists at most one \mathcal{Y} such that

$$[\alpha^{dec(\mathcal{Y})}\gamma] \oplus \mathbf{b} = (0,\dots,0) . \tag{12}$$

If $[\alpha^{dec(\mathcal{Y})}\gamma] \oplus \mathbf{b} \neq (0,\ldots,0)$, then

$$||([\alpha^{dec(\mathcal{Y})}\gamma] \oplus \mathbf{b}) \cdot \mathcal{X}^T \oplus \widetilde{c}_{\mathcal{Y}}|| = 2^{h-1}.$$

Hence,

$$N_{\widetilde{F}} \ge N_F + 2^{h-1} \left((2^h - 1) - 1 \right) .$$

Corollary 4.1. Suppose that there exists a balanced vectorial Boolean function $F(x_1, \ldots, x_h) = (f_1, \ldots, f_m)$ with nonlinearity N_F for $m \leq h$. Then there exists a balanced vectorial Boolean function \widetilde{F} with 2^sh input variables such that

$$N_{\widetilde{F}} \ge N_F + 2^{2^s h - 1} - \frac{1}{2} (2^{2^{s-1}h} + 2^{2^{s-2}h} + \dots + 2^{2h} + 2 \cdot 2^h)$$
.

Finally, we can obtain the following corollary from Corollary 4.1 and Proposition 4.1.

Corollary 4.2. If $n = 2^s h$, then there exists a balanced vectorial Boolean function $F(x_1, \ldots, x_n) = (f_1, \ldots, f_m)$ such that $m \le h$ and

$$N_F \geq \begin{cases} 2^{2^sh-1} - \frac{1}{2}(2^{2^{s-1}h} + 2^{2^{s-2}h} + \dots + 2^h + 2^{\frac{h}{2}+1}) & \text{if h is even} \\ 2^{2^sh-1} - \frac{1}{2}(2^{2^{s-1}h} + 2^{2^{s-2}h} + \dots + 2^h + 2^{\frac{h+1}{2}}) & \text{if h is odd} \end{cases}.$$

(Remark) Corollary 4.2 gives larger nonlinearity than Proposition 4.1 for $s \ge 1$ which corresponds to $2 \le m \le n/2$.

References

- J.F. Dillon. Elementary Hadamard difference sets. In The Sixth Southeastern Conference on Combinatorics, Graph Theory and Computing, pages 237–249, 1975.
- 2. C.Ding, G.Xiao and W.Shan. The stability theory of stream ciphers. Lecture Notes in Computer Science 561, Springer-Verlag, 1991. 20, 21
- H. Dobbertin. Construction of bent functions and balanced Boolean functions with high nonlinearity. In Fast Software Encryption, volume 1008 of Lecture Notes in Computer Science, pages 61–74, Springer-Verlag, 1995. 20, 21, 25
- T. Jakobsen and L. R. Knudsen. The interpolation attack on block ciphers. In Fast Software Encryption, volume 1267 of Lecture Notes in Computer Science, pages 28–40, Springer-Verlag, 1997.
- F.J. MacWilliams and N.J.A. Sloane. The theory of error-correcting codes. North-Holland Publishing Company, 1977. 20, 21, 22, 24

- M. Matsui. Linear cryptanalysis method for DES cipher. In Advances in Cryptology —EUROCRYPT '93 Proceedings, volume 765 of Lecture Notes in Computer Science, pages 386-397, Springer-Verlag, 1994.
 20, 21
- K. Nyberg. Perfect nonlinear S-boxes. In Advances in Cryptology —EUROCRYPT
 '91 Proceedings, volume 547 of Lecture Notes in Computer Science, pages 378–386,
 Springer-Verlag, 1991. 20, 21, 22
- 8. K. Nyberg. On the construction of highly nonlinear permutations. In Advances in Cryptology —EUROCRYPT '92 Proceedings, volume 658 of Lecture Notes in Computer Science, pages 92–98, Springer-Verlag, 1993. 20, 21, 25
- 9. O. S. Rothaus. On bent functions. In *Journal of Combinatorial Theory (A)*, 20:300–305, 1976. 20, 21
- J. Seberry, X.M. Zhang and Y. Zheng. Nonlinearity and propagation characteristics of balanced Boolean functions. In *Information and Computation*, 119(1):1–13, May 1995. 20, 21, 25
- X.M. Zhang and Y. Zheng. Cryptographically resilient functions. In IEEE Transactions on Information Theory, 43(5):1740–1747, September 1997.
- 12. T.Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. In *IEEE Transactions on Information Theory*, 30(5):776-780, 1984. 20

Equivalent Keys of HPC

Carl D'Halluin, Gert Bijnens, Bart Preneel*, and Vincent Rijmen**

Katholieke Universiteit Leuven, Dept. Electrical Engineering-ESAT/COSIC
K. Mercierlaan 94, B-3001 Heverlee, Belgium
{carl.dhalluin,gert.bijnens}@esat.kuleuven.ac.be
{bart.preneel,vincent.rijmen}@esat.kuleuven.ac.be

Abstract. This paper presents a weakness in the key schedule of the AES candidate HPC (Hasty Pudding Cipher). It is shown that for the HPC version with a 128-bit key, 1 in 256 keys is weak in the sense that it has 2^{30} equivalent keys. An efficient algorithm is proposed to construct these weak keys and the corresponding equivalent keys. If a weak key is used, it can be recovered by exhaustive search trying only 2^{89} keys on average. This is an improvement by a factor of 2^{38} over a normal exhaustive key search, which requires on average 2^{127} attempts. The weakness also implies that HPC cannot be used in standard constructions for hash functions based on block ciphers. The analysis is extended to HPC with a 192-bit key and a 256-bit key, with similar results. For some other key lengths, all keys are shown to be weak. An example of this is the HPC variant with a 56-bit user key and block length of 128 bits, which can be broken in 2^{31} attempts on average.

1 Introduction

The AES candidate HPC [3] is a block cipher with a variable block length and a variable algorithm: depending on the required block length range, five different versions are defined. In this paper we only look at the version called HPC Medium (sub-cipher number 3), which is the version that supports a block length of 128 bits, as required for the AES candidates. We focus on the HPC expanded key generation. A user key of 128, 192, or 256 bits is expanded to a KX-table containing 256 64-bit words, or 16 384 bits. The ciphertext only depends on the plaintext, the spice¹, and the KX-table. In this paper we assume the spice to be known. Thus if two different user keys K_1 and K_2 generate the same KX-table, then $\mathrm{HPC}_{K_1}(P) = \mathrm{HPC}_{K_2}(P)$ for every plaintext P. Keys K_1 and K_2 are called equivalent keys. In the specifications of HPC [4] it is stated:

"Two keys are equivalent if they expand to the same key-expansion table. The likelihood is negligible for keys of size < 1/2 the key-expansion table

^{*} F.W.O. Research Associate, sponsored by the Fund for Scientific Research - Flanders (Belgium).

^{**} F.W.O. Postdoctoral Researcher, sponsored by the Fund for Scientific Research - Flanders (Belgium).

¹ The *spice* can be regarded as a second key that need not be concealed.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 29–42, 1999. © Springer-Verlag Berlin Heidelberg 1999

size, 8192 bits. For keys longer than this, some will be equivalent, but there is no feasible way to discover an equivalent key pair."

David Wagner announced at the 2nd AES Conference [6] that HPC has equivalent keys based on the generic structure of the key expansion (see Sect. 2.3); he did not analyze their structure. For 128-bit user keys, we have discovered that exactly 2^{-8} of the keys each have 2^{30} equivalent keys. In this paper these keys are called weak keys. For 192-bit user keys, approximately 2^{-7} of the keys each have 2^{42} or 2^{32} equivalent keys, and approximately 2^{-16} of the keys each have 2^{74} equivalent keys. For 256-bit user keys, approximately 2^{-16} of the keys each have 2^{64} or 2^{66} equivalent keys and approximately 2^{-16} of the keys each have 2^{64} or 2^{66} equivalent keys and approximately 2^{-24} of the keys each have 2^{98} equivalent keys. These results are summarized in Table 1, and are proven in Sections 4 and 6.

Key length	# weak keys	# equivalent keys
128	2^{120}	2^{30}
192	2^{184}	2^{42}
192	2^{184}	2^{32}
192	2^{176}	2^{72}
256	2^{249}	2^{32}
256	2^{248}	2^{34}
256	2^{241}	2^{66}
256	2^{240}	2^{64}
256	2^{232}	2^{98}

Table 1. Approximate number of weak keys and number of equivalent keys

In Sect. 2 we explain how the expanded key table is calculated. Section 3 clarifies how the key expansion results in equivalent keys. In Sect. 4 we compute the number of weak keys and we present a method to construct weak keys. The impact of weak keys on exhaustive key search is treated in Sect. 5. In Sect. 6 we briefly search for weak keys if a user key with length other than 128 bits is used. In Sect. 7 we show that HPC-based hash functions are insecure, and in Sect. 8 we discuss how the weak keys can be avoided. Finally we conclude in Sect. 9.

In this paper all numbers in hexadecimal notation (indicated with a subscript x) are written with the least significant byte on the right side. Numbers in decimal notation have no subscript or a subscript d. The least significant bit (that is, the rightmost bit) is numbered bit 0.

2 HPC Expanded Key Generation

The expanded key table (denoted by KX-table) contains 286 64-bit words. The KX-table depends on the user key and on the sub-cipher number sc. The last 30

entries of the KX-table are equal to the first 30:

$$KX[i+256] = KX[i]$$
 for $i = 0, 1, ..., 29$.

This means that the KX-table effectively contains $256 \cdot 64 = 16384$ bits.

The KX-table is calculated in four steps. Firstly the table is filled with 256 pseudo-random values. Secondly the user key is XORed into the table. The goal of the *stirring function* is to make all the 256 entries of the table depend on the user key. Finally the last 30 entries of the table are set equal to the first 30. We discuss these steps in more detail below.

2.1 Filling the KX-table with Pseudo-random Values

The first entries of the KX-table are initialized using three mathematical constants (with sc denoting the sub-cipher number):

$$KX[0] = PI19 + sc \tag{1}$$

$$KX[1] = E19 * the key length$$
 (2)

$$KX[2] = R220$$
 rotated left over sc bits (3)

where PI19 = 3141592653589793238_d , E19 = 2718281828459045235_d , R220 = 14142135623730950488_d , sc = 3, and the key length is 128, 192, or 256.

Now the remaining 253 words of the table are pseudo-randomly filled with the following equation for $i=3,\ldots,255$, where '~' denotes the XOR operation and '>>', '<<' denote right and left shift operations respectively.

$$KX[i] = KX[i-1] + (KX[i-2] ^ KX[i-3] >> 23 ^ KX[i-3] << 41)$$
. (4)

2.2 XORing the User Key into the KX-table

The user key is XORed into the first entries of the KX-table. If the user key K has 128 bits, we have $K = K_H \parallel K_L$ with K_H the most significant 64 bits of K, and K_L the least significant 64 bits of K_L t

$$KX[O] \stackrel{\sim}{=} K_L$$

 $KX[1] \stackrel{\sim}{=} K_H$.

If the length of the user key is 192 bits or 256 bits, then we have to XOR the appropriate part of the key to KX[2] and KX[3] as well.

2.3 Stirring Function

The whole KX-table is made key dependent by means of the iterative stirring function. This function has eight internal state variables s0 to s7, that are initialized with s0 = KX[248], ..., s7 = KX[255]. The stirring function is run three times

Fig. 1. Pseudo-code (C notation) for the stirring function (the numbers are written in decimal notation)

```
for (j=0; j<3; j++)
                           /* Number of passes is 3
                                                               */
  for (i=0; i<256; i++) { /* Run over the entire KX-table */
        s0 = (KX[i] \cdot KX[(i+83)\&255]) + KX[s0\&255]
 (1)
 (2)
        s1 += s0
 (3)
        s3 ^= s2
 (4)
        s5 -= s4
 (5)
        s7 ^= s6
 (6)
        s3 += s0>>13
 (7)
        s4 ^= s1<<11
 (8)
        s5 ^= s3<<(s1&31)
 (9)
        s6 += s2>>17
        s7 |= s3+s4
(10)
        s2 -= s5
(11)
        s0 -= s6^i
(12)
        s1 ^= s5 + PI19
(13)
        s2 += s7>>j
(14)
(15)
        s2 ^= s1
        s4 -= s3
(16)
(17)
        s6 ^= s5
(18)
        s0 += s7
(19)
        KX[i] = s2 + s6
        }
```

(j-loop) over the entire KX-table (i-loop), allowing each bit to influence every other bit. The code for the stirring function is given in Fig. 1.

We denote the set of eight state variables before pass i, j of the stirring function by $\mathbf{s_{i,j}}(K)$ where K is the user key. We denote the stirring function of pass i, j by $F_{i,j,K}$. Hence the initial internal state is $\mathbf{s_{0,0}}(K)$. Note that $\mathbf{s_{0,0}}(K)$ does not depend on the user key K. The internal state after the stirring function is denoted by $\mathbf{s_{final}}(K)$. Thus we have

$$\begin{split} \mathbf{s_{final}}(K) &= F_{255,2,K}(\mathbf{s_{255,2}}(K)) \\ &= F_{255,2,K}(F_{254,2,K}(\mathbf{s_{254,2}}(K))) \\ &= F_{255,2,K}(F_{254,2,K}(\cdots(F_{0,2,K}(\mathbf{s_{0,2}}(K))))) \\ \mathbf{s_{0,2}}(K) &= F_{255,1,K}(\mathbf{s_{255,1}}(K)) \ . \end{split}$$

This leads to the following iterated definition:

$$\mathbf{s_{i,j}}(K) = F_{i-1,j,K}(\mathbf{s_{i-1,j}}(K)) \text{ for } i = 1, 2, \dots, 255 \text{ and } j = 0, 1, 2,$$

 $\mathbf{s_{0,j}}(K) = F_{255,j-1,K}(\mathbf{s_{255,j-1}}(K)) \text{ for } j = 1, 2.$

If we consider the stirring function for the HPC variant with 128-bit user key, we note that only $F_{i,j,K}$, where $0 \le i \le 1$ and $0 \le j \le 2$, depend on the user

key. Thus if $\mathbf{s_{i_0,j_0}}(K_1) = \mathbf{s_{i_0,j_0}}(K_2)$ for two different user keys K_1 and K_2 and $i_0 \geq 1$, we know that $\mathbf{s_{i,j_0}}(K_1) = \mathbf{s_{i,j_0}}(K_2)$ for $i_0 \leq i \leq 255$, and we also know that $\mathbf{s_{0,j_0+1}}(K_1) = \mathbf{s_{0,j_0+1}}(K_2)$ if $j_0 = 0,1$ or that $\mathbf{s_{final}}(K_1) = \mathbf{s_{final}}(K_2)$ if $j_0 = 2$. We can even control $F_{0,j,K}$ and $F_{1,j,K}$ independently, since $F_{0,j,K}$ only depends on the least significant half of the user key, and $F_{1,j,K}$ only depends on the most significant half.

These considerations show that if we can find a set of user keys $\{K_a\}$ such that $F_{1,j,K_a}(\mathbf{s_{1,j}}(K_a))$ and $F_{0,j,K_a}(\mathbf{s_{0,j}}(K_a))$ are constant over the entire set $\{K_a\}$ and only depend on j, then the KX-table will also be constant for all user keys in the set $\{K_a\}$. This means that we can define an equivalence relationship in the set of all user keys; the condition for equivalence is: "expands into the same KX-table as." The disjunct set $\{K_a\}$ is called an equivalence class.

In this paper we will show that such equivalence classes do exist. For the 128-bit variant of HPC, we can find 2^{90} equivalence classes, each containing 2^{30} elements.

3 Equivalent Keys (Key Length Equal to 128 Bits)

Before the stirring function is applied to the KX-table, the user key is XORed into the KX-table (see Sect. 2.2). For the AES candidate HPC with user key length equal to 128 bits, only KX[0] and KX[1] are influenced by the user key. The other values KX[2] up to KX[255] are independent of the 128-bit user key. Of course this is only true *before* the stirring function is applied to the KX-table.

A closer investigation of the key expansion leads us to an equation, from now on called the *dangerous equation*:

$$s0 = (KX[i] \cdot KX[(i+83)\&255]) + KX[s0\&255]$$
. (5)

This equation is dangerous because when $s0 \& 255_d = i$, different values of KX[i] can produce the same result. We investigate this problem closer for i = 0 and i = 1.

3.1 Dangerous Equation for i = 0

First we check the dangerous equation (5) for i=0. The initial values of s0 to s7 are constants for key length 128 bits and sub-cipher number 3, and are equal to the initial values of respectively KX [248] to KX [255]. Their values are shown in Table 2. The initial value of KX [0] is equal to (PI19+sub-cipher number) ^ K_L. Thus we have

$$KX[0] = 2b992ddf \ a23249d9_x \ ^K_L \ .$$

Since so & $255_d = 4b_x = 75_d$, the dangerous equation for i = 0 becomes:

$$s0 = (KX[0] \cdot KX[83]) + KX[75]$$
.

The values KX[83] and KX[75] are constants (for key length equal to 128 and sub-cipher number equal to 3) and are equal to

```
{\tt KX[83]} = {\tt 093817dc} \ {\tt b93586e6}_x \ , \\ {\tt KX[75]} = {\tt 989a3714} \ {\tt d85dee74}_x \ .
```

If we evaluate the dangerous equation numerically, we find

```
s0 ^= (22a13a03 1b07cf3f_x ^ {\rm K_L}) + 989a3714 d85dee74_x .
```

This mapping from K_L to s0 is clearly a bijection, hence a modification of K_L induces a modification of s0. The state variables s0 to s7 all change with high probability due to the stirring function (see Sect. 2.3) and the whole KX-table changes significantly. Hence we cannot find equivalent keys that differ in the least significant 64 bits of the user key.

Table 2. Initial state for the stirring function of HPC with user key length equal to 128 bits (all values in hexadecimal notation)

```
\begin{array}{|c|c|c|c|c|c|c|}\hline {\rm so} & 4{\rm cfd66f0} & 5{\rm ab4064b}_x\\ {\rm s1} & 6{\rm f7d4e0e} & 4107{\rm bd8c}_x\\ {\rm s2} & {\rm eadadb90} & 0{\rm f4b3d2a}_x\\ {\rm s3} & {\rm f24cb427} & {\rm cb159a63}_x\\ {\rm s4} & {\rm d7ee7776} & {\rm c0ecbc0b}_x\\ {\rm s5} & 3{\rm c255969} & 3{\rm f8f7688}_x\\ {\rm s6} & 390009{\rm fb} & 99146{\rm a25}_x\\ {\rm s7} & 1{\rm e5d58c2} & 7{\rm c76052e}_x\\ \end{array}
```

3.2 Dangerous Equation for i = 1

If $\mathtt{K_L}$ is constant, then the state variables after the first pass of the i-loop of the stirring function (i=0) are constant as well. For i=1, the initial state of $\mathtt{s0}$ is part of the internal state of the stirring function, after one pass of the i-loop (i=0). For now we will assume that we can choose $\mathtt{KX}[0]$ such that the least significant byte of $\mathtt{s0}$ after the first pass of the i-loop (i=0) of the stirring function is equal to $\mathtt{01}_x$ (see Sect. 4). The dangerous equation (5) for the case i=1 becomes:

```
s0 = (KX[1] \cdot KX[84]) + KX[1].
```

Let T denote ($KX[1] ^ KX[84]$) + KX[1]. The value KX[84] is completely determined by the key length (128) and the sub-cipher number (3). We have

$$KX[84] = a8f07353 9f208716_x$$
.

If bit ω' of KX[84] is set to 1, then it does not matter whether bit ω' of KX[1] is set to 0 or 1. This can easily be shown by looking at the individual bits of T. The

bit on position t is denoted by a subscript t, with t = 0 for the least significant bit. We obtain for $0 \le t \le 63$:

```
\begin{split} \mathbf{T}_t &= \mathbf{KX} [\mathbf{1}]_t ~ \hat{ } ~ \mathbf{KX} [\mathbf{84}]_t ~ \hat{ } ~ \mathbf{KX} [\mathbf{1}]_t ~ \hat{ } ~ c_t \\ &= \mathbf{KX} [\mathbf{84}]_t ~ \hat{ } ~ c_t ~ , \\ c_{t+1} &= (\mathbf{KX} [\mathbf{1}]_t ~ \hat{ } ~ \mathbf{KX} [\mathbf{84}]_t) \mathbf{KX} [\mathbf{1}]_t ~ \hat{ } ~ (\mathbf{KX} [\mathbf{1}]_t ~ \hat{ } ~ \mathbf{KX} [\mathbf{84}]_t ~ \hat{ } ~ \mathbf{KX} [\mathbf{1}]_t) c_t \\ &= (\mathbf{KX} [\mathbf{1}]_t ~ \hat{ } ~ \mathbf{KX} [\mathbf{84}]_t) \mathbf{KX} [\mathbf{1}]_t ~ \hat{ } ~ \mathbf{KX} [\mathbf{84}]_t c_t ~ , \end{split}
```

with c_{t+1} the carry bit generated at bit position t and $c_0 = 0$. Investigation of these equations leads us to three observations:

- 1. T_t does not depend on $KX[1]_t$;
- 2. c_{t+1} does not depend on $KX[1]_t$ if and only if $KX[84]_t = 1$;
- 3. c_{t+1} is not used for t = 63, since we work with 64-bit quantities.

The support of KX[84] (set of bit positions on which we have a 1), is denoted by Ω' . Examining the numerical value of KX[84], we find that $\Omega' = supp(\text{KX}[84]) = \{1, 2, 4, 8, 9, 10, 15, 21, 24, 25, 26, 27, 28, 31, 32, 33, 36, 38, 40, 41, 44, 45, 46, 52, 53, 54, 55, 59, 61, 63 \}. We can thus complement the value of any combination of bits on positions <math>\omega' \in \Omega' \cup \{63\}$ of KX[1] without changing the value of $T = (\text{KX}[1]^KX[84]) + \text{KX}[1]$. Since the set $\Omega' \cup \{63\}$ contains 30 elements, we can choose 30 bits of KX[1] at random, without changing the value of the right half of the dangerous equation. As KX[1] = dca375e0 59b0b980_x ^ K_H, we can easily find a set of 2^{30} equivalent keys, by complementing the bit positions ω' of K_H.

Now we have shown that if we can set the least significant byte of $\mathfrak{s0}$ equal to $\mathfrak{01}_x$ (after one pass of the *i*-loop (i=0) of the stirring function), then we can construct 2^{30} equivalent keys by simply complementing the bits on a subset of 30 specific bit positions ω' of K_H . In the next section it is shown that the least significant byte of $\mathfrak{s0}$ can be set equal to $\mathfrak{01}_x$ by choosing specifically bits 0 to 7 and bits 13 to 20 of K_L .

If we look at a 128-bit weak key, then the set of bit positions ω that can be complemented, is equal to $\Omega = \{\omega \mid \omega = \omega' + 64, \text{ with } \omega' \in \Omega'\} \cup \{127\}.$

4 Counting Weak Keys and a Construction Method

We start by analyzing the first pass of the *i*-loop of the stirring function (i=0). We use a specific notation to calculate the value of the least significant byte of $\mathfrak{s0}$ after the 19 steps: $\mathfrak{s0}_n$ indicates the value of $\mathfrak{s0}$ before step n. $\mathfrak{s0}_1$ is the initial value of $\mathfrak{s0}$ (before the stirring function is applied). $\mathfrak{s0}_{\mathrm{end}}$ is the value of $\mathfrak{s0}$ after the first pass of the stirring function for i=0. In this calculation we only have to keep track of the least significant byte of $\mathfrak{s0}$. If a certain operation does not affect this byte, the operation is not taken into account (e.g., addition with $\mathfrak{s1}_7 \ll 11$). We substitute the values of Table 2 in the equations. We have:

$$s0_{end} = s0_{18} + s7_{18}$$
,

which can be reworked (see Fig. 1) to:

$$s0_{end} = s0_2 + 36_x + (0b_x | (54_x + s0_2 \gg 13))$$
 (6)

As shown in Sect. 3, we have a weak key if the least significant byte of $\mathfrak{s0}_{\mathrm{end}}$ is equal to $\mathfrak{01}_x$. If we take a look at (6) we see the OR operation (denoted by '|') with the fixed value $\mathfrak{0b}_x$, which forces the least significant 4 bits of the result to be either \mathfrak{b}_x or \mathfrak{f}_x . If we want $\mathfrak{s0}_{\mathrm{end}}$ to be $\mathfrak{01}_x$, we must put restrictions on the least significant 4 bits of $\mathfrak{s0}_2 + 3\mathfrak{6}_x$. This can be translated to the following condition on K_L :

$$K_L \mod 10_x = 8 \text{ or } K_L \mod 10_x = c_x$$
.

Appendix A provides further details on the construction of the weak keys. Some examples are given in Table 3.

Table 3. Examples of weak keys of the 128-bit variant of HPC (all values in hexadecimal notation)

5 Exhaustive Key Search for User Key Length 128 Bits

If we know that someone uses a weak key, we can find that key by exhaustive key search, trying on average 2^{89} different key values. We only have to construct every weak key using the procedure given in Appendix A and check whether it is the correct key. In this procedure we can choose freely 120 bits of the user key. Since the values of 30 bits of K_H do not influence the KX-array, we can assign arbitrary values to these bits. Hence only 120-30=90 key bits remain to be recovered. This is a major improvement compared to a brute force attack in which we have to recover 128 bits (or 120 bits if we know that we have a weak key).

Even if we do not know whether a key is weak, an exhaustive key search can be improved by starting the search with weak keys. In 1 case out of 256 the user key will be weak, which implies that the search will be successful after at most 2^{90} encryptions.

6 HPC with Other Key Lengths

The key length influences the KX-table due through the initial value of KX[1], see (2), and the recursion formula (4). We briefly study the AES candidate HPC with key length equal to 192 and 256 bits. The results are summarized in Table 1. Finally we take a short look at some other key lengths.

6.1 Key Length 192 Bits

Since the number of key words increases from two to three, we evaluate the dangerous equation (5) three times:

$$i = 0$$
: s0 ^= (KX[0] ^ KX[83]) + KX[s0&255] (7)

$$i = 1 : s0 ^= (KX[1] ^KX[84]) + KX[s0\&255]$$
 (8)

$$i = 2$$
: s0 ^= (KX[2] ^ KX[85]) + KX[s0&255]. (9)

As before, it turns out that initially, $\mathfrak{so}\&255 \neq 0$, hence there are no weak keys for i=0. In the cases i=1 and i=2 however, \mathfrak{so} depends on some key bits (see Sect. 4). This allows us to generate two sets of 2^{184} weak keys (together a fraction of approximately $2^{-7} - 2^{-16}$ of the key space) with a different number of equivalent keys. These two sets have an intersection containing approximately 2^{172} weak keys ($\approx 2^{-16}$ of the key space) with 2^{74} equivalent keys each.

- The case i=1 is the same as for 128-bit keys. Hence, we know that 1 key in 256 is weak. We also know that the number of equivalent keys is determined by the Hamming weight of KX[84], and the most significant bit of KX[84]. Calculations lead us to KX[84] = efc64dbb cb9f7b71_x with Hamming weight 42. The most significant bit is set to 1. This means that each weak key has 2^{42} equivalent keys. Two equivalent keys in this class differ only in the middle 64 bits. If one knows in advance that someone uses a weak key, the key is found by exhaustive search after on average 2^{141} attempts.
- For the case i=2 we also know that 1 key in 256 is weak. Since KX[85] = 8842f3d7 13b09bab_x with Hamming weight 32 and most significant bit equal to 1, each weak key has 2^{32} equivalent keys. Two equivalent keys differ only in the most significant 64 bits.
- Simulations show that we can combine the two previous cases². We find that approximately 2^{-16} of the user keys are weak, corresponding to 2^{74} equivalent keys each. Two equivalent keys can differ in the most significant 128 bits.

6.2 Key Length 256 Bits

For key length equal to 256 bits, we have to evaluate the dangerous equation (5) for i = 1, i = 2, and i = 3. Now we can generate three sets of 2^{248} weak keys with

² There is no reason why these two events should be independent. Computer simulations show however that this assumption results in a reasonable approximation.

a different number of equivalent keys. The intersections between these three sets are non-empty and contain weak keys with a significant number of equivalent keys.

- For the case i = 1 we have $\texttt{KX}[84] = \texttt{e8e10c4d} \ \texttt{d1eb6c1d}_x$ with Hamming weight 32 and the most significant bit is set to 1. Hence 1 in 256 user keys is weak, with 2^{32} equivalent keys.
- For the case i=2 we have $\texttt{KX}[85] = \texttt{6cd2af08} \ 790165f3_x$ with Hamming weight 31 but the most significant bit is set to 0. Hence 1 in 256 user keys is weak, with 2^{32} equivalent keys.
- For the case i=3 we have KX[86] = ec7833a4 9d9bce38_x with Hamming weight 34 and the most significant bit is set to 1. Hence 1 in 256 user keys is weak, with 2^{34} equivalent keys.
- We can combine the previous three cases, and generate four other sets of weak keys with a different number of equivalent keys. An overview is given in Table 1. There are approximately An example of a weak key with approximately 2^{232} weak keys with 2^{98} equivalent keys; an example of such a key is $K = [K_3 \parallel K_2 \parallel K_1 \parallel K_0]$, where K_i denote 64-bit quantities, and

$$[\mathbf{K}_3\parallel\mathbf{K}_2] = \mathtt{aabe6c6d} \ \mathtt{e3a06a02} \ \mathtt{b7650b34} \ \mathtt{6c73cf94}_x \ , \qquad (10)$$

6.3 Other Key Lengths

If we keep the sub-cipher number constant, the entries of the KX-array before the application of the stirring function depend on the key length only (see equations (2) and (4)). If we can find key lengths such that KX [248] & $255_d = 00_x$, then evaluating the dangerous equation (5) for i = 0 gives

$$s0 = (KX[0] \cdot KX[83]) + KX[0]$$

which means that all keys are weak keys. Table 4 shows the first 10 key lengths for which all the keys are weak. Their number of equivalent keys is also shown. It is an interesting coincidence that for a key length of 56 bits (as for the DES [1]), all keys are weak keys with 2^{24} equivalent keys. This means that the user key can be recovered by exhaustive key search, after on average 2^{31} attempts. For the sake of clarity we repeat that we are studying the HPC version with 128-bit block length. The version with 64-bit block lengths, as the DES, has a different sub-cipher number and has no weak keys.

7 HPC-Based Hash Functions

The existence of such a large number of equivalent keys has a serious impact on the use of HPC in the standard constructions for hash functions based on block ciphers. The problems of weak keys in this context have already been discussed earlier, see for example [2].

Key length	KX[83]		# Equivalent keys
56	87d0e495	$0b884dc5_x$	2^{24}
403	387d8891	$\mathtt{8dbf8aa7}_x$	2^{34}
608	3ff84d03	$8\mathrm{d}713835_x$	2^{33}
1190	d5d15fd2	$86e68311_x$	2^{32}
1993	3d16d709	$7971336a_x$	2^{34}
2491	1029f33c	$\mathtt{e3daa470}_{x}$	2^{31}
2512	64b16068	$745 {\tt df8ea}_x$	2^{32}
2656	593f5ca5	e7b6a6ad $_x$	2^{39}
2983	fa06ddd3	$\mathtt{f052af40}_x$	
3245	f28fb98c	$7b26832f_x$	2^{35}

Table 4. Key lengths for which all the keys are weak, and their number of equivalent keys

The building block of most hash functions based on block ciphers is known as the Davies-Meyer hash function (although the authorship of this function is uncertain). This is an iterated hash function; in step i, the chaining variable H_{i-1} and the ith message block X_i are compressed to the next value of the chaining variable H_i as follows:

$$H_i = E_{X_i}(H_{i-1}) \hat{H}_{i-1}$$
,

where $E_K()$ denotes encryption with a block cipher E using the key K. This mapping is denoted the *compression function*.

A first observation is that equivalent keys for the block cipher E lead to trivial collisions for the hash function. Indeed, if the X_i 's are chosen from a single equivalence class (that contains 2^{30} keys), the value of H_i will not change. A second observation is that equivalent keys lead to trivial 2nd preimage attacks in a similar way. The fact that in 1 case out of 256 key search can be sped up with a factor of 2^{38} implies that finding a 2nd preimage can take advantage of the same speed-up.

By applying an affine transformation of variables (for example, swapping X_i and H_{i-1}), the above attacks may become attacks on the compression function rather than on the hash function. Nevertheless, a strong compression function is a desirable criterion (see [2] for more details on the relation between weak keys and the strength of the compression function).

8 How to Eliminate the Weak Keys

R. Schroeppel has announced a 'tweak' of HPC that should eliminate the weak keys [5]. The following line of code is added to the stirring function after line (1) (see Fig. 1).

(1a)
$$s2 += KX[i]$$

This ensures indeed that changes in the user key are propagated to the entire state, which implies that with very high probability no two 'short' user keys will result in the same KX table. An alternative solution would be to create part of the KX table as a bijection of the user key. This would guarantee that for user keys with up to 16 384 bits, no two keys are equivalent.

9 Conclusion

In this paper we have discussed a serious weakness of the key expansion of AES candidate HPC. For the 128-bit user key version, it is shown that exactly 2^{-8} of the user keys are weak keys, each with 2^{30} equivalent keys. These weak keys and their corresponding equivalent keys can be constructed using a very simple and efficient algorithm. Such a weak key can be found by exhaustive key search, trying only 2^{90} different keys.

We investigated the presence of weak keys for user key length 192 and 256 bits, and showed that respectively 2^{-7} and $1.5 \cdot 2^{-7}$ of the user keys are weak keys. We even found key lengths, for which all user keys are weak. Finally we note that HPC in its present form is not suitable for use in hash function constructions.

Acknowledgment

The authors would like to thank R. Schroeppel for motivating their research by announcing an attractive prize for the best cryptanalysis of HPC.

References

- FIPS 46, "Data encryption standard," NBS, U.S. Department of Commerce, Washington D.C., Jan. 1977.
- B. Preneel, R. Govaerts, J. Vandewalle, "Hash functions based on block ciphers: a synthetic approach," Advances in Cryptology, Proceedings Crypto'93, LNCS 773, D. Stinson, Ed., Springer-Verlag, 1994, pp. 368–378. 38, 39
- R. Schroeppel, "An overview of the Hasty Pudding Cipher," AES-submission, http://www.cs.arizona.edu/rcs/hpc, 1998.
- 4. R. Schroeppel, "The Hasty Pudding Cipher: Specific NIST requirements," AES-submission, 1998. 29
- 5. R. Schroeppel, "Tweaking the Hasty Pudding Cipher," http://www.cs.arizona.edu/ rcs/hpc/tweak, 1999. 39
- D. Wagner, "Equivalent keys for HPC," rump session talk at the 2nd AES Conference, Rome (I), March 22-23, 1999, http://www.cs.berkeley.edu/ãaw 30

A Construction of 128-bit Weak Keys

In this appendix we show how to construct the weak 128-bit user keys. The construction of 192-bit and 256-bit weak keys is omitted due to space restrictions.

To construct a 128-bit weak key we have to satisfy (6) for the least significant byte of $\mathfrak{s0}_2$:

$$01_{x} = s0_{2} + 36_{x} + 0b_{x} | (54_{x} + s0_{2} \gg 13)$$

$$\updownarrow$$

$$L = 0b_{x} | R , \qquad (13)$$

where $L = cb_x - s0_2$ and $R = (54_x + s0_2 \gg 13)$. In order to satisfy this equation, bits 0, 1, and 3 of L must be 1. In Sect. 4 it is proven that in order to set these bits to 1, we have to choose $K_L \& f_x$ equal to 8_x or c_x . A weak key can be constructed as follows:

- Assign a random value to bits 2, 4, 5, 6 and 7 of K_L . Set bits 0 and 1 of K_L equal to 0, and set bit 3 of K_L equal to 1.
- Calculate the least significant byte of $s0_2 = s0_1^{(KX[0] ^ KX[83])} + KX[75] = 4b_x^{(d9_x^K_L^e6_x)} + 74_x.$
- Calculate the byte value $L = cb_x sO_2 = Ob_x | R$. Then calculate R

$$\begin{split} \mathbf{R} &= \mathbf{54}_x + \bigg(\mathbf{s0_1} \hat{\ } \big((\mathbf{KX[0]} \hat{\ } \mathbf{KX[83]}) + \mathbf{KX[75]} \bigg) \bigg) \!\gg\! 13 \\ &= \mathbf{54}_x + \bigg(\Big(\mathbf{s0_1} \!\gg\! 13 \Big) \hat{\ } \big((\mathbf{KX[0]} \hat{\ } \mathbf{KX[83]}) \!\gg\! 13 + \mathbf{KX[75]} \!\gg\! 13 + p \Big) \bigg) \\ &= \mathbf{54}_x + \bigg(\mathbf{a0}_x \hat{\ } \bigg(\Big(\mathbf{92}_x \hat{\ } (\mathbf{K_L} \!\gg\! 13) \hat{\ } \mathbf{ac}_x \bigg) + \mathbf{ef}_x + p \bigg) \bigg) \ , \end{split}$$

where p is a carry bit, which we have to calculate first.

- To calculate p, choose at random bits 8 to 12 of K_L and calculate the carry bit p of ((KX[0] ^ KX[83]) + KX[75]). The bit p=1 if and only if the value of this expression, calculated only with the 13 least significant bits of KX[0], KX[83] and KX[75], is larger than $2^{13}-1$. In the other case p=0. Hence the choice of bits 0 to 12 of K_L determines the value of p.
- Now we have to solve the equation $L=0b_x|R$. First assign a random value to bits 0, 1 and 3 of R. The unknown values in this equation are bits 13 to 20 of K_L . Now we can solve for $K_L\gg 13$, and we find exactly one solution.
- Now we have a value for bits 0 to 20 of K_L . These 21 bits are completely determined by choosing 13 random bits (bits 2, 4, 5, 6, 7 and 8 to 12 of K_L , and bits 0, 1 and 3 of R).
- We can choose the other 64-21=43 bits of K_L and the 64 bits of K_H at random. We now have obtained a weak key. This means that the number of weak keys is exactly 2^{120} (since we chose freely 120 bits to construct the weak key). This means that exactly 2^{-8} of the keys are weak keys. In Sect. 3 it is proven that each weak key has 2^{30} equivalent keys.

Example

We now construct a weak key, using the techniques described in the previous section.

- We assign the bit value 0 to bits 2, 4, 5, 6 and 7 of K_L . We set bit 0 and 1 of K_L equal to 0, and set bit 3 of K_L equal to 1. Then the least significant byte of the key has the value 08_x .
- We calculate the least significant byte of \mathfrak{so}_2 . This gives \mathfrak{eo}_x .
- We set bits 8 to 12 of K_L to 0. It turns out that p=1.
- We calculate $L = cb_x sO_2 = eb_x$. We set bits 0, 1 and 3 of R to 0. We solve the equation $R = eO_x$. It turns out that $(K_L \gg 13) = OO_x$.
- The other bits 21–63 of K_L and 0–63 of K_H are set 0. Now we have constructed the weak key

$$\mathbf{K} = [\mathbf{K_H} \parallel \mathbf{K_L}] = \mathbf{00000000} \ \mathbf{00000000} \ \mathbf{00000000} \ \mathbf{00004008}_x \ .$$

It is easy to check that this key is indeed a weak key with 2^{30} equivalent keys.

Cryptanalysis of Five Rounds of CRYPTON Using Impossible Differentials

Haruki Seki¹ and Toshinobu Kaneko²

¹ TAO (Telecommunications Advancement Organization of Japan)
1-1-32 Shin'urashima-cho, Kanagawa-ku, Yokohama, 221-0031 Japan
hseki@yokohama.tao.go.jp
² Science University of Tokyo
2641 Yamazaki, Noda-shi, Chiba, 278-8510 Japan
kaneko@ee.noda.sut.ac.jp

Abstract. An block cipher CRYPTON based on the structure of SQUARE is a candidate algorithm for the AES. Recently Lim changes the S-box construction and key scheduling, and suggested modified version(version 1.0) in FSE'99. In this paper we present an attack on CRYPTON reduced to 5 rounds. This attack is based on impossible differentials[7]. 4 rounds of CRYPTON has impossible differential, we use this to show that CRYPTON version 1.0 reduced to 5 rounds can be attacked using 2^{83.4} chosen plaintext and ciphertext pairs. This attack can be also applied to CRYPTON version 0.5 using less chosen plaintext and ciphertext pairs.

1 Introduction

C.H.Lim proposed an block cipher CRYPTON[1] based on the structure of SQUARE[5]. It is a candidate algorithm for the AES. Several analyses were proposed to this cipher. Weak keys are discovered in CRYPTON version 0.5[3]. In[4], G.Bijnens applied higher order differential attack to 6 rounds of CRYPTON. To overcome some weakness, recently Lim suggested modified version (version 1.0)[2]. This new version changes the S-box construction and key scheduling.

In this paper we applied a variant of differential cryptanalysis, which is called impossible differential cryptanalysis, to CRYPTON reduced to 5 rounds. The idea of cryptanalysis with impossible differentials was applied to DES S-boxes by E.Biham[6]. Recently Skipjack reduced to 31 rounds was attacked by cryptanalysis with impossible differentials[7]. It seems that this attack is powerful for some ciphers with impossible differentials. Both version of CRYPTON has impossible differential, which ensure that for all keys there are no pairs of inputs with particular differences with the property that after 4 rounds of encryption the outputs have some other particular differences. This impossible differential is applied to attack on CRYPTON reduced to 5 rounds. Using $2^{83.4}$ chosen plaintext and ciphertext pairs, fifth roundkey of 128 bits of CRYPTON version 1.0 can be obtained. CRYPTON version 0.5 can be also attacked using $2^{75.6}$ chosen plaintext and ciphertext pairs.

This paper is organized as follows. Section 2 gives a preliminary. Section 3 briefly reviews algorithms of CRYPTON. In section 4 we describe a 4-round impossible differential of CRYPTON. In section 5 we discuss the attack on CRYPTON version 1.0 reduced to 5 rounds. In section 6 we discuss the attack on CRYPTON version 0.5 reduced to 5 rounds. We conclude in section 7.

2 Preliminary

In this paper we use next definitions.

Definition 1. $A^i_{\gamma}, A^i_{\pi}, A^i_{\tau}, A^i_{\sigma}$: Input of $\gamma, \pi, \tau, \sigma$ transformation in round i

Definition 2. $B^i_{\gamma}, B^i_{\pi}, B^i_{\tau}, B^i_{\sigma}$: Output of $\gamma, \pi, \tau, \sigma$ transformation in round i

Definition 3. A': Differential value of a pair of A

Definition 4. P, C: Plaintext, ciphertext

Definition 5. K_e^i : A 128-bit roundkey in round i

Definition 6. A[i][j]: A 8-bit word of *i*-th row and *j*-th column of 4×4 matrix

3 Description of CRYPTON

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]

Fig. 1. Byte coordinate of CRYPTON

The block cipher CRYPTON is designed based on SQUARE. The data is arranged to a 4×4 byte array as shown in Fig.1. CRYPTON uses next transformation in each round.

- Nonlinear byte substitution γ : Two different transformations γ_o, γ_e are used alternatively in successive rounds. γ_o is used in odd rounds, γ_e is used in even rounds. This transformation consists of byte-wise substitutions.
- Linear columnwise bit permutation π : Two different transformation π_e, π_o are used. π_e is used in even rounds, π_o is used in odd rounds. These transformations calculates two bits at once by exoring the value of two bits in corresponding positions in three different bytes of the column. They can be implemented using four mask bytes, denoted $m_0 = fc_x, m_1 = f3_x, m_2 =$ $cf_x, m_3 = 3f_x.$

$$B_{\pi_o}[i][j] = \bigoplus_{k=0}^3 \left((A[k][j]) \wedge m_{(i+j+k)mod\ 4} \right) \quad \text{ for odd rounds }.$$

$$B_{\pi_e}[i][j] = \bigoplus_{k=0}^{3} \left((A[k][j]) \wedge m_{(i+j+k+2) \bmod 4} \right) \quad \text{for even rounds} \ .$$

For even rounds of CRYPTON version 0.5 $m_{(i+j+k+2)mod\ 4}$ is replaced by $m_{(i+j+k+1)mod\ 4}$.

- Linear column-to-row transposition τ : This operation simply rearranges 4×4 byte array by moving the byte at the (i, j)-th position to the (j, i)-th position.
- Key addition $\sigma_{K_e^i}$: This operation is xoring data with *i*-th roundkey K_e^i of 128 bits.

The encryption round functions are defined for odd and even rounds as follows.

-
$$\rho_{oK_e^i}(A) = (\sigma_{K_e^i} \circ \tau \circ \pi_o \circ \gamma_o)(A)$$
 for odd rounds
- $\rho_{eK_e^i}(A) = (\sigma_{K_e^i} \circ \tau \circ \pi_e \circ \gamma_e)(A)$ for even rounds

And linear output transformation ϕ_e is used at last round.

-
$$\phi_e = \tau \circ \pi_e \circ \tau$$

Then encryption of full 12-round CRYPTON is described as

-
$$Enc = \phi_e \circ \rho_{eK_0^{12}} \circ \rho_{oK_0^{11}} \circ \cdots \circ \rho_{eK_0^2} \circ \rho_{oK_0^1} \circ \sigma_{K_0^0}$$

We don't consider the use of ϕ_e in our attack, because this function uses only known quantities.

A 4-Round Impossible Differential 4

In this section we show CRYPTON has a 4-round impossible differential. Fig. 2 describes one pattern of an impossible differential.

An Impossible Differential: Given an input of the form (a), then there cannot be pairs $B_{\gamma}^{4\prime}$ of the form (b) after byte substitution γ in round 4.

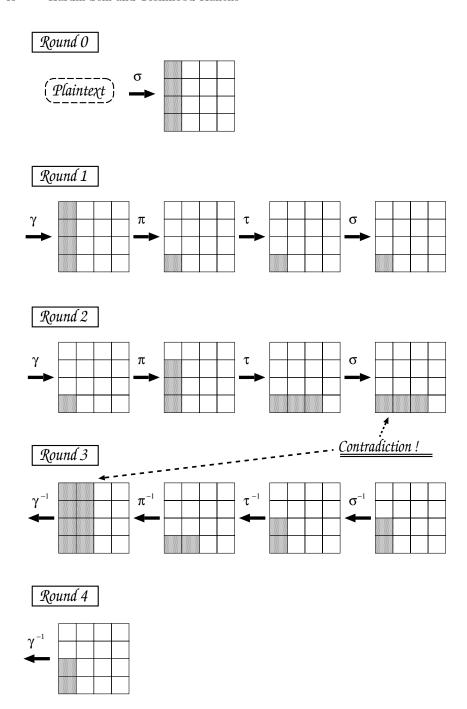


Fig. 2. The 4-round impossible differential. White squares represents zero differences, gray squares represents nonzero differences.

(a) The differences of 4 words in only one column of the plaintext pair are nonzero, and the differences of all 12 words in other 3 columns are zero. For example,

$$P' = \begin{bmatrix} a & 0 & 0 & 0 \\ b & 0 & 0 & 0 \\ c & 0 & 0 & 0 \\ d & 0 & 0 & 0 \end{bmatrix}$$

(b) The differences of one or two words in any one column of $B_{\gamma}^{4\prime}$ are nonzero, and those of all other 15 or 14 words are zero. For example,

$$B_{\gamma}^{4\prime} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ e & 0 & 0 & 0 \\ f & 0 & 0 & 0 \end{bmatrix}$$

To prove this, we make the following observations: (see Fig. 2).

- 1. If $B_{\gamma}^{4\prime}$ has the form stated in (b), then all 8 words of 2 columns of $B_{\gamma}^{3\prime}$, which is the difference after γ transformation in round 3, are zero(for example, column 0 and 1 in Fig. 2). So all 8 words of 2 columns of $B_{\sigma}^{2\prime}$, which is the difference after σ transformation in round 2, are zero.
- 2. If the difference of plaintext pair has the form stated in (a), then some words of only one row of $B_{\tau}^{1\prime}$, which is the difference after τ transformation in round 1, are nonzero. 3 or 4 words of any column of $B_{\pi}^{2\prime}$, which is the difference after π transformation in round 2, are nonzero. ¹ So the difference of 3 or 4 words of at least one row of $B_{\sigma}^{2\prime}$, i.e., 3 or 4 columns, are nonzero. 3. From 1 just 2 columns of $B_{\sigma}^{2\prime}$ are zero. From 2 at least 3 columns of $B_{\sigma}^{2\prime}$ are
- nonzero. This is a contradiction.

An Attack on CRYPTON Version 1.0 Reduced to 5 5 Rounds

In this section we describe cryptanalysis of CRYPTON version 1.0 reduced to 5 rounds. The attack is based on the 4-round impossible differential with additional one round at the end. An attack is as follows.

1. Choose structure of 2^{32} plaintexts which differ at four words of only column 3, i.e., P[0][3], P[1][3], P[2][3], P[3][3], having all the possible values in it. Such structure proposes about 2⁶³ pairs of plaintexts(see Fig.2).

 $^{^1}$ π transformation always transform one nonzero differential word into 3 or 4 nonzero differential words.

Fig. 3. Ciphertext pattern used in an attack. The impossible differentials in round 4 always satisfy $B_{\gamma}^{5\prime}$ pattern used.

2. Given $2^{51.4}$ structures ($2^{83.4}$ plaintexts), we calculate B_{γ}^5 using linear transformation of ciphertexs C as follows.

$$B_{\gamma}^5 = \pi_o \left(\tau(C) \right) . \tag{1}$$

- 3. We collect all those pairs which differ only at four words of the *i*-th row of B_{γ}^{5} , which are pairs after γ transformation in round 5(see Fig.3). Only pairs, which has nonzero difference in 1 or 2 words of *i*-th column of $B_{\gamma}^{4\prime}$ and zero differences in all other 14 or 15 words, satisfy this $B_{\gamma}^{5\prime}$. By this operation on average one structure proposes $2^{-33} (= 2^{-96} \times 2^{63})$ such pairs, and thus only about $2^{18.4} (= 2^{-33} \times 2^{51.4})$ pairs remain.
- 4. We decrypt remaining pairs with all possible 32-bit value of *i*-th row of K_{eq}^5 . The decryption is expressed as follows.

$$(A_{\gamma}^{5}[i][0] , A_{\gamma}^{5}[i][1], A_{\gamma}^{5}[i][2], A_{\gamma}^{5}[i][3])$$

$$= (\gamma_{o}(B_{\gamma}^{5}[i][0] \oplus K_{eq}^{5}[i][0]), \gamma_{o}(B_{\gamma}^{5}[i][1] \oplus K_{eq}^{5}[i][1])$$

$$, \gamma_{o}(B_{\gamma}^{5}[i][2] \oplus K_{eq}^{5}[i][2]), \gamma_{o}(B_{\gamma}^{5}[i][3] \oplus K_{eq}^{5}[i][3])) . \tag{2}$$

Where we express equivalent key K_{eq}^5 of round key K_e^5 as follows.

$$K_{eq}^5 = \pi_o \left(\tau(K_e^5) \right) . \tag{3}$$

5. Next we calculate the difference of the *i*-th column of $B_{\gamma}^{4\prime}$ as follows.

$$(B_{\gamma}^{4\prime}[0][i] , B_{\gamma}^{4\prime}[1][i], B_{\gamma}^{4\prime}[2][i], B_{\gamma}^{4\prime}[3][i])^{t}$$

$$= \left(\pi_{e}\left(\tau(A_{\gamma}^{5\prime}[i][0], A_{\gamma}^{5\prime}[i][1], A_{\gamma}^{5\prime}[i][2], A_{\gamma}^{5\prime}[i][3])\right)\right)^{t}.$$
(4)

As we know that such a difference as those of the form (b) is impossible, every key that proposes such a difference is a wrong key. For each pair we try all the 2^{32} possible values of the *i*-th row of equivalent key K_{eq}^5 , and verify whether the decrypted values have the form (b). It is expected that about 6×2^{16} values proposed this difference, and thus we are guaranteed that these 6×2^{16} values are not the correct equivalent key of round 5. After analyzing the $2^{18.4}$ pairs, there remain only about $2^{32} \times (1 - 6 \times 2^{-16})^{2^{18.4}} = 2^{-14}$ wrong values of the equivalent key of round 5. It is thus expected that only one value remains, and this value must be the correct 32-bit equivalent key of *i*-th row of K_{eq}^5 .

6. If we do above procedure for i = 0, 1, 2, 3, then 32-bit equivalent key of ith row of K_{eq}^5 can be obtained independently. Finally 128-bit round key of round 5 is obtained by linear transformation as follows.

$$K_e^5 = \tau \circ \pi_o \circ K_{eq}^5 \ . \tag{5}$$

The time complexity of recovering 128-bit roundkey of round 5 is equivalent to about $2^{83.4}$ $\pi \circ \tau$ transformation and 2^{43} encryptions. ²

6 An Attack on CRYPTON Version 0.5 Reduced to 5 Rounds

The procedure of an attack on CRYPTON version 0.5 is the same as that on version 1.0. The number of plaintext and ciphertext pairs needed for an attack is less than that needed on version 1.0.

We can collect pairs which have nonzero differences only in 4 words of *i*-th row of $B_{\gamma}^{5\prime}$ with higher probability than the case of version 1.0. This is caused by the difference in the S-box construction. We explain this as follows (see appendix for details).

- 1. When 4 words of only the third column of P' are nonzero, we can collect pairs which have nonzero difference in 4 words of only the first row of $B_{\gamma}^{5\prime}$ with probability $2^{-87.2}$. From the same plaintexts we can also collect pairs which have nonzero difference in 4 words of only the third row of $B_{\gamma}^{5\prime}$ with probability $2^{-87.2}$.
- 2. Similarly when 4 words of only the 0-th column of P' are nonzero, we can collect pairs which have nonzero difference in 4 words of only the 0-th row (and only the second row) of $B_{\gamma}^{5\prime}$ with probability $2^{-87.2}$.

So we can obtain the first row and the third row of K_{eq}^{5} using $2^{42.6}$ structures ($2^{74.6}$ plaintexts) as shown in 1, and 0-th row and the second row of K_{eq}^{5} using $2^{42.6}$ structures ($2^{74.6}$ plaintexts) as shown in 2.3 Totally $2^{75.6}$ plaintext and ciphertext pairs are needed for an attack.

 $[\]begin{array}{l} ^2 \ 2^{32} + 2^{32} \times (1-6 \times 2^{-16}) + 2^{32} (1-6 \times 2^{-16})^2 \cdots + 2^{32} \times (1-6 \times 2^{-16})^{2^{18.4}} = 1.3 \times 2^{45} \\ \pi \circ \tau \circ \gamma \circ \sigma_{K_e} \ \ \text{computations are needed. Since 5 round encryption consists of 5} \\ \text{applications of } \pi \circ \tau \circ \gamma \circ \sigma_{K_e}, \text{ this time complexity is equal to about } 2^{43} \ \ \text{encryptions.} \\ ^3 \ 2^{42.6} \times 2^{63} \times 2^{-87.2} = 2^{18.4} \ \ \text{pairs remain for each case as shown in } 1 \ \text{or } 2. \end{array}$

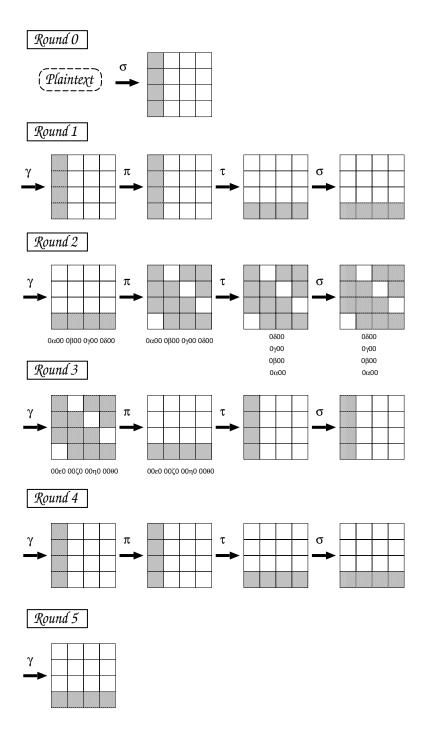


Fig. 4. One of differential patterns of CRYPTON version 0.5 used for an attack

7 Conclusion

In this paper we described an attack on CRYPTON reduced to 5 rounds using impossible differential on 4-round CRYPTON. To obtain 128-bit roundkey of round 5 of CRYPTON version 1.0, we need $2^{83.4}$ chosen plaintext and ciphertext pairs. This attack can be also applied to CRYPTON version 0.5 using $2^{75.6}$ chosen plaintext and ciphertext pairs.

References

- 1. C.H.Lim., "http://www.nist.gov/aes" 43
- C.H.Lim., "A Revised Version of CRYPTON: CRYPTON Version 1.0," Fast Software Encryption, 1999, pp. 31-46.
- 3. S.Vaudenay., "Weak keys in CRYPTON," announcement on NIST's electronic AES forum, http://www.nist.gov/aes. 43
- 4. C.D'Halluin, G.Bijnens, V.Rijimen, and B.Preneel., "Attack on Six Rounds of CRYPTON," Fast Software Encryption, 1999, pp.47-60. 43
- J.Daemen, L.Knudsen and V.Rijmen, "The block cipher Square," Fast Software Encryption, 1997, Spring-Verlag, LNCS 1267, pp.149-165.
- E.Biham, A.Shamir., "Differential Cryptanalysis of DES-like Cryptosystems," CRYPTO'90 Proceedings, Spring-Verlag, 1990, pp.2-21. 43
- 7. E.Biham, A.Biryukov, and A.Shamir., "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials," EUROCRYPT'99 Proceedings, Spring-Verlag, LNCS 1952, 1999, pp.12-23. 43

Appendix

We explain the case of 1 in section 6. When only the third column of P' is nonzero, we can collect pairs which have nonzero difference of 4 words of only the third row of $B_{\gamma}^{5\prime}$ with probability $2^{-87.2}$. Fig.4 shows this differential pattern. 4 words of $B_{\gamma}^{2\prime}$ with nonzero difference have next form.

$$\left(B_{\gamma}^{2\prime}[3][0],B_{\gamma}^{2\prime}[3][1],B_{\gamma}^{2\prime}[3][2],B_{\gamma}^{2\prime}[3][3]\right)=\left(0\delta00,0\gamma00,0\beta00,0\alpha00\right)\;. \tag{6}$$

 $0\alpha00$ is 8-bit word which has nonzero difference on 4-5 bits. After π , τ and σ transformation in round 2, $A_{\gamma}^{3\prime}$ has the form shown in Fig.4. When every 3 words of each column of $B_{\gamma}^{3\prime}$ has the same value as $00\varepsilon0$, after π transformation in round 3 $B_{\pi}^{3\prime}$ has nonzero differences of 4 words of only the third row. The probability of γ transformation in round 2 is $2^{-24} \left(= (2^{-6})^4\right)$. The probability of γ transformation in round 3 is $2^{-63.2} \left(= (\frac{5}{3} \times 2^{-6})^{12}\right)$. So the probability of $B_{\gamma}^{5\prime}$ having nonzero difference of 4 words of only third row is $2^{-87.2}$. From the same plaintexts we can also collect $B_{\gamma}^{5\prime}$ having nonzero difference of 4 words of only first row with probability $2^{-87.2}$. The case of 2 in section 6 can be also explained by the similar procedure as above.

Cryptanalysis of Two Cryptosystems Based on Group Actions

Simon R. Blackburn* and Steven Galbraith**

Department of Mathematics, Royal Holloway, University of London, Egham, Surrey TW20 0EX, United Kingdom {s.blackburn,s.galbraith}@rhbnc.ac.uk

Abstract. The paper cryptanalyses two public key cryptosystems based on $SL_2(\mathbb{Z})$ that have been recently proposed by Yamamura.

1 Introduction

Yamamura [13,14] has recently described two public-key cryptosystems based on subsemigroups of $SL_2(\mathbb{Z})$. This paper cryptanalyses both of these systems. We show that a plaintext may be efficiently obtained from the corresponding ciphertext and public key, and hence both systems are insecure.

There have been other proposals for cryptographic primitives based on group theory. A public key cryptosystem based on 'logarithmic signatures' in finite groups was proposed by Qu and Vanstone [9]. This system was cryptanalysed by Blackburn, Murphy and Stern [1,2]. Related work, for example by Magliveras and Memon [6], investigated the suitability of a cryptosystem based on permutation groups. A hybrid system, primarily based on a knapsack problem but also involving logarithmic signatures, was proposed by Qu and Vanstone [10] and was cryptanalysed by Nguyen and Stern [7]. Tillich and Zémor [12] proposed a hash function based on $SL_2(\mathbb{F}_{2^n})$. Geiselmann [5] described how to find collisions for this hash function; see Charnes and Pieprzyk [3] for other comments on this scheme.

The basic cryptanalytic approach of this paper is as follows. The ciphertext in the cryptosystem described in [14] is a complex number z. It turns out that it is easy to derive the first bit of the plaintext from z. An example of the possible ciphertexts output by the cipher is given in Figure 2. These complex numbers clearly fall into two easily distinguished regions in the complex plane, depending on the first bit of the plaintext. Once the first bit has been recovered, it is easy to compute the ciphertext corresponding to the plaintext with the first bit removed. Hence we may recover all the plaintext one bit at a time.

The bulk of this paper shows that the phenomenon illustrated in Figure 2 occurs for all possible choices of private key, and so the cryptosystem proposed in [14] is insecure. We also show how to reduce the security of the cryptosystem

^{*} This author is supported by an E.P.S.R.C. Advanced Fellowship

^{**} This author thanks the E.P.S.R.C. for their support

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 52–61, 1999. © Springer-Verlag Berlin Heidelberg 1999

proposed in [13] to the security of the cryptosystem proposed in [14]; hence both cryptosystems are insecure.

The rest of this paper is organised as follows. Section 2 contains the background on $\mathrm{SL}_2(\mathbb{Z})$ that we require. Section 3 describes the two cryptosystems that Yamamura proposes. Section 4 cryptanalyses these systems, and Section 5 discusses a slightly more general class of cryptosystems.

2 Background

Both the Yamamura cryptosystems are based on properties of the group $SL_2(\mathbb{Z})$ of 2×2 integer matrices of determinant 1 under multiplication. This section summarises properties of this group that we will need.

Define matrices $A, B \in \mathrm{SL}_2(\mathbb{Z})$ by

$$A = \begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix} \qquad \text{and} \qquad B = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

It is well known that these matrices generate $\mathrm{SL}_2(\mathbb{Z})$. It is easy to check that $A^3=B^2=-I$, where I is the 2×2 identity matrix. In fact, it is possible to characterise $\mathrm{SL}_2(\mathbb{Z})$ in terms of abstract group theory as an 'amalgamated free product' of the cylic groups of order 6 and 4 generated by A and B respectively; see Robinson [11, Section 6.4] for any facts about amalgamated free products that we use.

The theory of amalgamated free products shows that each element $g \in \mathrm{SL}_2(\mathbb{Z})$ has a unique representation as an element in 'normal form'. More precisely, there exists a unique non-negative integer n, a unique $\epsilon \in \{I, -I\}$ and unique elements $s_1, s_2, \ldots s_n \in \{A, A^2, B\}$ such that

$$g = \epsilon s_1 s_2 \cdots s_n \tag{1}$$

and such that for all $k \in \{1, 2, ..., n-1\}$ either:

$$-s_k \in \{A, A^2\}$$
 and $s_{k+1} = B$, or $-s_k = B$ and $s_{k+1} \in \{A, A^2\}$.

We now introduce some more geometrical notions. We write $GL_2(\mathbb{C})$ for the group of all 2×2 matrices with complex entries and non-zero determinant. This group acts on $\mathbb{C} \cup \{\infty\}$ by associating

$$g = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{GL}_2(\mathbb{C})$$

with the 'Möbius transformation' defined by

$$z + -(az + b)/(cz + d)$$
.

It is well known — see Jones and Singerman [4] — that Möbius transformations map circles to circles (we include the limiting case of the lines in our definition of circle).

Since $SL_2(\mathbb{Z})$ is a subgroup of $GL_2(\mathbb{C})$, we may associate each element of $SL_2(\mathbb{Z})$ with a Möbius transformation. The Möbius transformations associated with $SL_2(\mathbb{Z})$ actually preserve the upper half plane $\mathcal{H} = \{z \in \mathbb{C} : Im(z) > 0\}$.

As Yamamura observes, this action gives rise to an efficient algorithm to compute the normal form (1) of an arbitrary element $g \in SL_2(\mathbb{Z})$, up to sign; we may describe this algorithm as follows. Define regions O, P, Q and R of \mathcal{H} by

$$\begin{split} O &= \{z \in \mathcal{H} : |z| > 1, |\text{Re}(z)| \le 1/2\}, \\ P &= \{z \in \mathcal{H} : |z| \ge 1, |\text{Re}(z)| \ge 1/2\}, \\ Q &= \{z \in \mathcal{H} : |z| \le 1, |z - 1| \le 1\} \text{ and } \\ R &= \mathcal{H} - (O \cup P \cup Q). \end{split}$$

These regions are depicted in Figure 1.

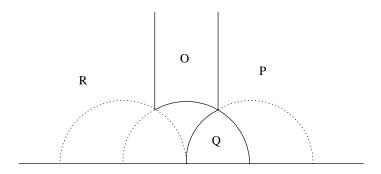


Fig. 1. The regions O, P, Q and R

(The region O is the standard fundamental domain for the action of $SL_2(\mathbb{Z})$ on \mathcal{H} .) Choose a point z_0 in the interior of O; for example $z_0 = 2i$. Define $z = g(z_0)$. The following algorithm finds the normal form of g, up to sign.

Algorithm 1

- 1. Set k = 0.
- 2. If $z \in O$ then halt.
- 3. Set k = k + 1.
- 4. If $z \in P$ then set $s_k = A$, if $z \in Q$ then set $s_k = A^2$ and if $z \in R$ then set $s_k = B$.
- 5. Set $z = s_k^{-1}(z)$.
- 6. Return to step 2.

The sequence $s_1s_2...s_n$ is the normal form of $\pm g$; see Yamamura [14] for details.

3 The Cryptosystems

This section describes the two public key cryptosystems proposed by Yamamura. We refer to the system proposed in [13] as the "polynomial-based scheme" and the system proposed in [14] as the "point-based scheme". We describe the two schemes in a different manner to Yamamura; we will prove below that the schemes we describe are no less general than the Yamamura schemes.

3.1 The Point-Based Scheme

A user generates its public key as follows. The user chooses words V_1 and V_2 in the generators A and B of $\mathrm{SL}_2(\mathbb{Z})$ so that V_1 and V_2 generate a free subsemigroup of $\mathrm{SL}_2(\mathbb{Z})$. This is done so that any word in V_1 and V_2 is in normal form with respect to A and B; moreover, one of V_1 and V_2 should not be an initial segment of the other. For example, for any integers i and j such that $i, j \geq 2$, a valid choice of V_1 and V_2 is $V_1 = (BA)^i$ and $V_2 = (BA^2)^j$. The user then chooses a matrix $M \in \mathrm{GL}_2(\mathbb{C})$ and a point p in the interior of the fundamental region O. The public key is defined to be an ordered triple (W_1, W_2, q) where $W_1, W_2 \in \mathrm{GL}_2(\mathbb{C})$ and $q \in \mathbb{C}$ are defined by

$$W_1 = M^{-1}V_1M,$$

 $W_2 = M^{-1}V_2M$
 $q = M^{-1}(p).$

The private key is the matrix M.

Encryption: Given a message of n values $i_1, i_2, \ldots, i_n \in \{1, 2\}$ the ciphertext is the point $q' = W(q) \in \mathbb{C}$ where

$$W = W_{i_1} W_{i_2} \cdots W_{i_n}.$$

Note that $M(q') = V_{i_1} V_{i_2} \cdots V_{i_n}(p)$.

Decryption: The receiver computes the point $p' = M(q') \in \mathcal{H}$. The receiver then uses Algorithm 1 to find the normal form of word $V_{i_1}V_{i_2}\cdots V_{i_n}$ in A and B. It is then easy to recover the sequence i_1, i_2, \ldots, i_n .

Following Yamamura, we ignore any practical issues relating to exact computation in \mathbb{C} . We assume that all calculations are performed to sufficient precision so that all the operations we consider may be carried out.

3.2 The Polynomial-Based Scheme

In this scheme, a user generates a public key as follows. The user chooses V_1 , V_2 and M as in the point-based scheme above. The user then chooses $a \in \mathbb{C}$ and a pair $F_1(x)$, $F_2(x)$ of 2×2 matrices over the polynomial ring $\mathbb{C}[x]$ such that $F_1(a) = V_1$ and $F_2(a) = V_2$. The public key is the pair $(W_1(x), W_2(x))$ where $W_i(x) = M^{-1}F_i(x)M$ for $i \in \{1, 2\}$. The secret key is the matrix M and the complex number a.

Encryption The sequence $i_1, i_2, \dots, i_n \in \{1, 2\}$ is encrypted as the matrix E(x) where

$$E(x) = W_2(x)W_1(x)^{i_1}W_2(x)W_1(x)^{i_2}\cdots W_2(x)W_1(x)^{i_n}W_2(x).$$

[One could have equally used the matrix $\prod_{j=1}^{n} W_{i_j}(x)$ as in the point-based scheme.]

Decryption The receiver calculates $g \in SL_2(\mathbb{Z})$, where $g = ME(a)M^{-1}$. The user chooses a point p in the interior of the fundamental domain O, calculates q = g(p) and recovers the message in essentially the same way as in the point-based scheme.

3.3 Some Comments

The description of the schemes above differ slightly from the presentation in Yamamura's papers [13,14]. Firstly, in the point-based scheme, Yamamura restricts M to lie in $GL_2(\mathbb{R})$, and so our scheme is more general in this respect. Secondly, in both schemes Yamamura allows a user to choose any generators $A_1, B_1 \in SL_2(\mathbb{Z})$ such that $A_1^3 = B_1^2 = -I$ in place of the matrices A and B. With this more general choice, our method of cryptanalysis still applies; see Section 5.

To avoid difficulties with the decryption method, Yamamura has recently [15] added the restriction that $A_1 = P^{-1}AP$ and $B_1 = P^{-1}BP$ for some matrix $P \in SL_2(\mathbb{Z})$. Note that, when A_1 and B_1 are chosen in this way, then an instance of Yamamura's scheme [13,14] with a matrix M is the same as an instance of the scheme described above with M replaced by PM. Hence the schemes above are as general as the schemes described by Yamamura if A_1 and B_1 are chosen in this manner.

4 Cryptanalysis

This section contains two subsections, which cryptanalyse each of the two cryptosystems in turn.

4.1 Cryptanalysis of the Point-Based Scheme

We begin this section with an example of a cryptanalysis. Suppose $V_1 = BABA$, $V_2 = BA^2$, p = 2i and

$$M = \begin{pmatrix} 0.8 - 0.3i & -0.2 + 0.4i \\ -0.8 + 0.9i & 2.7 + 0.3i \end{pmatrix} \in \mathrm{GL}_2(\mathbb{C}).$$

We will show that the points of \mathbb{C} corresponding to encryptions of messages with $i_1 = 1$ are easily distinguished from those with $i_1 = 2$. This is clear for our example: Figure 2 is a plot of all points in \mathbb{C} corresponding to messages of length between 1 and 9. The points fall into two regions depending on the bit i_1

of their corresponding plain texts: those points corresponding to messages with $i_1=1$ correspond precisely to the lower collection of points (those with negative imaginary part). Thus, given an intercepted ciphertext q', it is easy to determine the first bit i_1 of the corresponding plain text by finding which of the two regions q' lies in. Once i_1 has been determined, the first digit of the plain text may be stripped off by replacing q' by $V_{i_1}^{-1}(q')$ and the process repeated to determine earlier digits until q'=q.

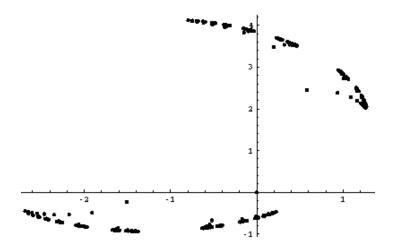


Fig. 2. Example of Ciphertexts

It remains to show that for all choices of parameters, the messages always fall into two regions in a similar way, and to show that these regions may be derived from the public key. We begin by showing that points corresponding to ciphertexts fall into two regions.

For $W_1, W_2 \in \mathrm{GL}_2(\mathbb{C})$ and $q \in \mathbb{C}$, define the set $S_q(W_1, W_2) \subseteq \mathbb{C}$ by

$$S_q(W_1,W_2) = \left\{W(q) \in \mathbb{C}: W = \prod_{j=1}^n W_{i_j} \text{ where } n \geq 0 \text{ and } i_j \in \{1,2\}\right\}.$$

Proposition 1. Let (W_1, W_2, q) be a public key of the point-based scheme. Then the sets $W_1S_q(W_1, W_2)$ and $W_2S_q(W_1, W_2)$ are separated by a boundary which consists of at most 3 circle segments.

Note: We include the limiting case of a line in our definition of 'circle'. Proof: Consider the words V_1 and V_2 , the point $p \in O$ and the matrix $M \in GL_2(\mathbb{C})$ that were used to construct the public key. Without loss of generality, the normal forms of V_1 and V_2 with respect to A and B may be written in the form

$$V_1 = \pm gAg_1$$
 and $V_2 = \pm gBg_2$,

where g, g_1 and g_2 are words in A and B. The fact that the normal form algorithm of Section 2 always works implies that $g^{-1}V_1S_p(V_1, V_2) \subseteq P \cup Q$ and $g^{-1}V_2S_p(V_1, V_2) \subseteq R$. The boundary of $P \cup Q$ consists of 3 circle segments. We may take the image of this boundary under $M^{-1}g$ as the boundary between $W_1S_q(W_1, W_2)$ and $W_2S_q(W_1, W_2)$. \square

In fact, the boundary that separates the two sets is usually much simpler than that constructed in the proposition. Indeed, except for the single case when $V_2 = gB$, we may take the image of the imaginary axis under $M^{-1}g$ as the boundary. Even in the exceptional case when $V_2 = gB$, the image of the imaginary axis separates the two sets once we remove single point $W_2(q)$. Hence for all practical purposes, we may assume that the two sets are separated by a circle.

It is now clear how cryptanalysis proceeds:

- 1. Generate some random points from $W_1S_q(W_1, W_2)$ and $W_2S_q(W_1, W_2)$ using the public key.
- 2. Find a circle that separates the two subsets that have been generated. Such a circle can be determined by a variety of methods.
- 3. For a ciphertext q', determine the first bit i_1 of the corresponding plaintext by determining which side of the circle q' lies.
- 4. Replace q' by $W_{i_1}^{-1}q'$ and, while $q \neq q'$, go to step 3.

4.2 Cryptanalysis of the Polynomial-Based Scheme

We now cryptanalyse the polynomial-based scheme, by reducing the problem to an instance of breaking the point-based scheme.

The cryptanalysis of the previous subsection makes use of the fact that $Mq \in O$. We justify why the methods will produce good results for all $q \in \mathbb{C}$ excluding a set of measure zero. We will show that all but a few points in $W_1S_q(W_1, W_2)$ and $W_2S_q(W_1, W_2)$ are separated by the boundary constructed in Proposition 1. This will allow us to use the same four steps above to recover almost all of the plaintext; see below.

Let $q \in \mathbb{C}$ and suppose that $p = M(q) \in \mathcal{H}$. (This is no real loss of generality, since a similar argument will work when $p \in -\mathcal{H}$.) Assume further that p is not on the boundary of any image of O under $\mathrm{SL}_2(\mathbb{Z})$. Then there is some $h \in \mathrm{SL}_2(\mathbb{Z})$ such that $h^{-1}(p)$ is in the interior of O. Let $i_1, \ldots, i_n \in \{1, 2\}$. The algorithm to find the normal form of an element of $\mathrm{SL}_2(\mathbb{Z})$ given by Yamamura works by considering the images of a point in the fundamental domain. Since our point p is in the image of the fundamental domain under the element h, the algorithm of Yamamura produces the normal form of $V_{i_1}V_{i_2}\cdots V_{i_n}h$ rather than of $V_{i_1}V_{i_2}\cdots V_{i_n}h$ when given $V_{i_1}V_{i_2}\cdots V_{i_n}(p)$. Now, the normal form of $V_{i_1}V_{i_2}\cdots V_{i_n}h$ almost always begins with the word V_{i_1} —the only way this can fail to happen is if h

begins with the normal form of $(V_{i_2}\cdots V_{i_n})^{-1}$, and this bad case is extremely unlikely. Indeed, the likelihood of the bad case occurring tends to 0 exponentially as n tends to infinity. Thus, if we construct our approximations to the sets $W_1S_q(W_1,W_2)$ and $W_2S_q(W_1,W_2)$ by sampling words in W_1 and W_2 of small length, the methods of the previous subsection will recover all but (at worst) the last few terms of the sequence i_1,i_2,\ldots,i_n with overwhelming probability. Even if problems occur with recovering the final few terms, this is easily overcome by exhaustive search.

We now cryptanalyse the polynomial-based scheme. We know that there is some $a \in \mathbb{C}$ such that the matrices $W_1(a)$ and $W_2(a)$ have determinant 1. Hence, $\gcd(\det(W_1(x)) - 1, \det(W_2(x)) - 1)$ is a polynomial which has a as a root. The roots of this polynomial may be computed to arbitrary precision by numerical methods; see Press $et\ al\ [8]$ for example. One of these roots will be the value of a we are seeking. In fact, for most choices of $F_1(x)$ and $F_2(x)$, there will only be one root a. For each candidate a' for a we repeat the following process.

Let E(x) be an intercepted ciphertext. Choose a point $p \in \mathbb{C}$ and compute p' = E(a')(p). We now use the method described in the cryptanalysis of the point-based scheme to express E(a') as a product of the matrices $W_1(a')$ and $W_2(a')$, thus giving us a candidate plaintext. If the plaintext encrypts to the intercepted ciphertext (which it is almost certain to do if a' = a), we have decrypted successfully.

5 More General Generators

It might be hoped that a different choice A_1 , B_1 of generators of $\mathrm{SL}_2(\mathbb{Z})$ such that $A_1^3 = B_1^2 = -I$ might resist the attacks above. (If this is done, a different decryption method must be found.) However, since we only care about A_1 and B_1 up to conjugation in $\mathrm{GL}_2(\mathbb{C})$, we may assume that A_1 and B_1 are of a restricted form (as outlined in the following proposition). We may then use this restricted form to show that no choices of A_1 and B_1 resist the attacks presented above.

Proposition 2. Let $A_1, B_1 \in SL_2(\mathbb{Z})$ be generators for $SL_2(\mathbb{Z})$ such that $A_1^3 = B_1^2 = -I$. Then there exists a matrix $N \in GL_2(\mathbb{R})$ such that

- (i) $N^{-1}A_1N = A \ and$
- (ii) the normal form (1) of $N^{-1}B_1N$ with respect to A and B has the property that $s_1 = s_n = B$.

Proof: The theory of amalgamated free products shows that any element of finite order in $SL_2(\mathbb{Z})$ is conjugate (by an element of $SL_2(\mathbb{Z})$) to an element in either the subgroup generated by A or the subgroup generated by B. Since A_1 has order 6, there exists $N \in SL_2(\mathbb{Z})$ that conjugates A_1 to either A or A^{-1} . Conjugating by the matrix T defined by

$$T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

preserves $\mathrm{SL}_2(\mathbb{Z})$ and maps A^{-1} to A. Thus, replacing N by NT if necessary, there exists $N \in \mathrm{GL}_2(\mathbb{R})$ such that property (i) holds and that $N^{-1}B_1N \in \mathrm{SL}_2(\mathbb{Z})$.

We may assume that the normal form (1) of $N^{-1}B_1N$ with respect to A and B has the property that $s_1 = B$, for if not we replace N by Ns_1 . Since $(N^{-1}B_1N)$ has finite order, the concatenation of two copies of the normal form of $N^{-1}B_1N$ cannot still be in normal form. Hence $s_n = B$ and so property (ii) holds. \Box

The form of A_1 and B_1 in Proposition 2 has the useful property that any word in normal form in A_1 and B_1 is also in normal form with respect to A and B once each occurrence of B_1 is replaced by the normal form of B_1 with respect to A and B. Moreover, two words in A_1 and B_1 have the property that one is not an initial segment of the other if and only if the same is true for their normal forms with respect to A and B. This means that the polynomial based scheme with general A_1 and B_1 is a special case of the scheme detailed above. Moreover, the point-based scheme with general A_1 and B_1 is also a special case of the scheme described in this paper, if we allow the point p to be an arbitrary point in \mathbb{C} . However, the methods discussed in the cryptanalysis of the polynomial-based scheme make the cryptosystem insecure in this case.

References

- S.R. Blackburn, S. Murphy and J. Stern, 'Weaknesses of a public-key cryptosystem based on factorizations of finite groups', in T.Helleseth (Ed) Advances in Cryptology
 — EUROCRYPT '93, Lecture Notes in Computer Science 765, Springer, Berlin, 1994, pp. 50-54. 52
- S.R. Blackburn, S. Murphy and J. Stern, 'The cryptanalysis of a public-key implementation of Finite Group Mappings', J. Cryptology Vol. 8 (1995), pp. 157-166.
- C. Charnes and J. Pieprzyk, 'Attacking the SL₂ hashing scheme', in J. Pieprzyk and R. Safavi-Naini (Eds) Advances in Cryptology ASIACRYPT '94, Lecture Notes in Computer Science 917, Springer, Berlin, 1995, pp. 322-330.
- 4. G. A. Jones, D. Singerman, Complex functions, Cambridge (1987) 53
- W. Geiselmann, 'A note on the hash function of Tillich and Zémor' in C.Boyd (Ed) Cryptography and Coding, Lecture Notes in Computer Science 1025, Springer, Berlin, 1995, pp.257-263.
- S.S. Magliveras and N.D. Memon, 'Algebraic properties of cryptosystem PGM', J. Cryptology, Vol. 5 (1992), pp. 167-183.
- P. Nguyen and J. Stern, 'Merkle-Hellman revisited: A cryptanalysis of the Qu-Vanstone cryptosystem based on group factorizations' in B.S. Kaliski (Ed) Advances in Cryptology — CRYPTO '97, Lecture Notes in Computer Science 1294, Springer, Berlin, 1997, pp. 198-212. 52
- W. Press, B. Flannery, S. Teukolsky and W. Vetterling, Numerical Recipes in C, 2nd Edition, Cambridge University Press, Cambridge, 1988.
- M. Qu and S.A. Vanstone, 'New public-key cryptosystems based on factorizations of finite groups', presented at AUSCRYPT '92.

- M. Qu and S.A. Vanstone, 'The knapsack problem in cryptography' in Finite fields: Theory, Applications, and Algorithms, Contemporary Mathematics Vol. 168, American Mathematical Society, 1994, pp. 291-308.
- D.J.S. Robinson, A Course in the Theory of Groups, Springer, New York, 1982.
- 12. J-P. Tillich and G. Zémor, 'Hashing with SL_2 ', in Y.G. Desmedt (Ed), *Advances in Cryptology CRYPTO '94*, Lecture Notes in Computer Science 839, Springer, Berlin, 1994, pp. 40-49. 52
- A. Yamamura, 'Public-key cryptosystems using the modular group', in Imai, Hideki (Eds) et al. International Workshop on the Theory and Practice of Cryptography, Lecture Notes in Computer Science 1431, Springer, Berlin, 1998, pp. 203-216. 52, 53, 55, 56
- 14. A. Yamamura, A functional cryptosystem using a group action, ACIPS to appear. 52, 53, 54, 55, 56
- 15. A. Yamamura, personal communication, 3 March 1999. 56

Probabilistic Higher Order Differential Attack and Higher Order Bent Functions

Tetsu Iwata and Kaoru Kurosawa

Department of Electrical and Electronic Engineering, Faculty of Engineering, Tokyo Institute of Technology 2-12-1 O-okayama, Meguro-ku, Tokyo 152-8552, Japan {tez, kurosawa}@ss.titech.ac.jp

Abstract. We first show that a Feistel type block cipher is broken if the round function is *approximated* by a low degree vectorial Boolean function. The proposed attack is a generalization of the higher order differential attack to a probabilistic one. We next introduce a notion of higher order bent functions in order to prevent our attack. We then show their explicit constructions.

1 Introduction

Consider a Feistel type block cipher with a round function G_K such that

$$(y_1, \dots, y_n) = G_K(x_1, \dots, x_n) \tag{1}$$

where K denotes a key. Then G_K can be viewed as a polynomial on $GF(2^n)$ or a set of Boolean functions $\{f_1, \ldots, f_n\}$ such that

$$y_i = f_i(x_1, ..., x_n) \text{ for } i = 1, ..., n$$
.

¿From a view point of polynomials, Jakobsen and Knudsen showed the interpolation attack which is effective if the degree of G_K is small [4]. Jakobsen further showed that the block cipher is broken even if G_K is approximated by a low degree polynomial [3].

On the other hand, from a view point of Boolean functions, Jakobsen and Knudsen showed the higher order differential attack [4]. It is effective if each of the degree of f_i is small, where the degree is defined as the degree of a Boolean function.

In this paper, we first show that the block cipher is broken even if each f_i is approximated by a low degree Boolean function. We call this attack a probabilistic higher order differential attack because our attack is a generalization of the higher order differential attack to a probabilistic one. (It can also be considered as a generalization of the differential attack [1] to a higher order one.)

We next introduce a notion of higher order bent functions in order to prevent our attack. Intuitively, an r-th order bent function is a Boolean function f such that $N_f^{(r)}$ is the maximum, where $N_f^{(r)}$ is defined as a distance from f to the set of Boolean functions with degree at most r. This means that an r-th order bent function is not approximated by any Boolean function with degree at most r if r is small.

We then present some explicit constructions of r-th order bent functions such that other cryptographic criteria are satisfied as well.

This paper is organized as follows. In Section 3, we review related works. In Section 4, we propose the probabilistic higher order differential attack. In Section 5, we introduce a notion of r-th order bent functions and show their explicit constructions.

2 Preliminaries

2.1 Notation

Consider a Feistel type block cipher with block size 2n and m rounds. Let $x = (x_L, x_R)$ denote the plaintext, where $x_L = (x_1, \ldots, x_n)$ and $x_R = (x_{n+1}, \ldots, x_{2n})$. Similarly, let $y = (y_L, y_R)$ denote the ciphertext. Let

$$C_0^L \stackrel{\triangle}{=} x_L$$
 and $C_0^R \stackrel{\triangle}{=} x_R$.

The round function G operates as follows.

$$\begin{cases} C_i^L = C_{i-1}^R , \\ C_i^R = G(k_i, C_{i-1}^R) \oplus C_{i-1}^L , \end{cases}$$

where k_i is a key of the *i*-th round. The ciphertext is given by

$$y = (y_L, y_R) = (C_m^R, C_m^L)$$
.

Further, we say that

$$(C_{m-1}^L, C_{m-1}^R) = E_K(x_L, x_R)$$

is the reduced cipher, where K is the key of the reduced cipher. Let $\tilde{y} = (\tilde{y}_L, \tilde{y}_R)$ denote the reduced ciphertext. That is,

$$\tilde{y} = (\tilde{y}_L, \tilde{y}_R) = (C_{m-1}^R, C_{m-1}^L)$$
.

In this paper, we assume that m is not large.

2.2 Degree of Boolean Functions

The degree of a Boolean function f, deg(f), is defined as the degree of the highest degree term of the algebraic normal form:

$$f(x_1, \dots, x_n) = a_0 \oplus \bigoplus_{1 \le i \le n} a_i x_i \oplus \bigoplus_{1 \le i < j \le n} a_{i,j} x_i x_j \oplus \dots \oplus a_{1,2,\dots,n} x_1 x_2 \cdots x_n .$$

The degree of a vectorial Boolean function $F(x_1, \ldots, x_n) = (f_1, \ldots, f_n)$ is defined as

 $\deg(F) \stackrel{\triangle}{=} \max_{i} \deg(f_i) .$

3 Related Works

3.1 Higher Order Differential Attack

The higher order differential attack [4] is based on the following proposition shown by Lai [5]. Let L_r denote an r-dimensional subspace of $GF(2)^n$.

Proposition 3.1. [5] Let f be a Boolean function. Then for any $w \in GF(2)^n$,

$$\bigoplus_{x \in L_{r+1}} f(x \oplus w) = 0$$

if and only if $\deg(f) \leq r$.

In a Feistel type block cipher, let x_R be kept constant. Then

$$\tilde{y}_R = F(x_L) ,$$

for some vectorial Boolean function F, where \tilde{y}_R is the right half of the reduced ciphertext. Suppose that $\deg(F) \leq r$ for any fixed x_R and any fixed key of the reduced cipher. Then the last round key k_m can successfully be recovered by using 2^{r+1} chosen plaintexts with average time complexity $2^r|K_m|$ by using Proposition 3.1, where K_m denotes the set of the last round keys.

3.2 Piling-Up Lemma

Matsui used the following lemma in the analysis of the linear attack [8].

Lemma 3.1 (Piling-up Lemma). For $a_i \in GF(2)$ with i = 1, ..., l, suppose that

$$\bigoplus_{1 \le i \le l} a_i = 0 .$$

Let a_i' be an independent random element of GF(2) such that $\Pr(a_i = a_i') \ge \mu$ for i = 1, ..., l. Then

$$\Pr(\bigoplus_{1 \le i \le l} a_i' = 0) \ge 1/2 + 2^{l-1}(\mu - 1/2)^l$$
.

4 Proposed Attack

In this section, we show that a Feistel type block cipher is broken even if the round function G is approximated by a low degree vectorial Boolean function. We call this attack the probabilistic higher order differential attack because our attack is a generalization of the higher order differential attack to a probabilistic one.

4.1 Algorithm of Our Attack

In a Feistel type block cipher with block size 2n, let x_R be kept constant. Then

$$\tilde{y}_R = F(x_L) \tag{2}$$

for some vectorial Boolean function F, where \tilde{y}_R is the right half of the reduced ciphertext. On the other hand, let G be the round function. Then

$$\tilde{y}_R = y_L \oplus G(k_m, y_R)$$
,

where $k_m \in K_m$ is the last round key and (y_L, y_R) is the ciphertext. Therefore,

$$\tilde{y}_R = F(x_L) = y_L \oplus G(k_m, y_R) . \tag{3}$$

Definition 4.1. We say that a vectorial Boolean function F(x) is (r, μ) -expressible if there exists a vectorial Boolean function F'(x) such that $\deg(F'(x)) \leq r$ and

$$\Pr_{x}(F(x) = F'(x)) \ge \mu .$$

Now suppose that $F(x_L)$ of eq.(2) is (r, μ) -expressible for any fixed x_R and any fixed key of the reduced cipher. Then the last round key $k_m \in K_m$ can be found by the proposed attack as shown below, where Algorithm 1 is used as a subroutine in Algorithm 2. Let K_m denote the set of the last round keys.

Step 1: Choose $x_R \in GF(2)^n$ randomly. Choose $w \in GF(2)^n$ and a full rank $(r+1) \times n$ matrix L over GF(2) randomly.

Step 2: For all $a \in GF(2)^{r+1}$, compute the ciphertext $y(a) = (y_L(a), y_R(a))$ of a plaintext $(aL \oplus w, x_R)$.

Step 3: For each $k_i \in K_m$, compute

$$\sigma = \bigoplus_{a \in GF(2)^{r+1}} y_L(a) \oplus G(k_i, y_R(a)) .$$

If $\sigma = (0, ..., 0)$, then let $u_i = 1$. Otherwise, let $u_i = 0$.

Fig.1. Algorithm 1

Step 1: Let $T_i = 0$ for $1 \le i \le |K_m|$.

Step 2: For j = 1, ..., N, do:

- (a) Run Algorithm 1.
- (b) For each $k_i \in K$, let $T_i = T_i + u_i$.

Step 3: Output k_c such that T_c is the maximum.

Fig.2. Algorithm 2

4.2 Analysis of Our Attack

The complexity of our attack is analyzed as follows.

Lemma 4.1. For $a_i \in GF(2)^n$ with i = 1, ..., l, suppose that

$$\bigoplus_{1 \le i \le l} a_i = (0, \dots, 0) .$$

Let a_i' be an independent random element of $GF(2)^n$ such that $Pr(a_i = a_i') \ge \mu$ for i = 1, ..., l. Then

$$\Pr(\bigoplus_{1 \le i \le l} a_i' = (0, \dots, 0)) \ge (1/2 + 2^{l-1}(\mu - 1/2)^l)^n$$
.

Proof. Denote the j-th bit of a_i as $a_{i,j}$ and the j-th bit of a_i' as $a_{i,j}'$ for $i=1,\ldots,l$ and $j=1,\ldots,n$. The equation $\bigoplus_{1\leq i\leq l}a_i'=(0,\ldots,0)$ holds if and only if

$$\bigoplus_{1 \le i \le l} a'_{i,j} = 0$$

holds for j = 1, ..., n. On the other hand, $\Pr(a_i = a_i) \ge \mu$ implies that

$$\Pr(a_{i,j} = a'_{i,j}) \ge \mu$$

for j = 1, ..., n. Then from Lemma 3.1, the result follows.

We assume that $\{F(aL \oplus w) \mid a = (0, \dots, 0), \dots, (1, \dots, 1)\}$ behaves as independent random 2^{r+1} vectors if L and w are chosen randomly, where L is a full rank $(r+1) \times n$ matrix over GF(2).

Theorem 4.1. Suppose that $F(x_L)$ of eq.(2) is (r, μ) -expressible for any fixed x_R and any fixed key of the reduced cipher. If μ is close to one, then the last round key can be found by using $N2^{r+1}$ chosen plaintexts with average time complexity $2^rN|K_m|$ and the success probability

$$\sum_{1 \le i \le N} \binom{N}{i} p^i (1-p)^{N-i} \left(\sum_{0 \le j \le i-1} \binom{N}{j} 2^{-nj} (1-2^{-n})^{N-j} \right)^{|K_m|-1},$$

where

$$p = 1 - 2^{r+1}n(1 - \mu)$$
.

Proof. Since $F(x_L)$ is (r, μ) -expressible, there exists a vectorial Boolean function F'(x) such that $\deg(F'(x)) \le r$ and

$$\Pr_{L,w}(F(aL \oplus w) = F'(aL \oplus w)) \ge \mu . \tag{4}$$

First, from Proposition 3.1, it holds that

$$\bigoplus_{a} F'(aL \oplus w) = (0, \dots, 0) . \tag{5}$$

On the other hand, at step 3 of Algorithm 1,

$$\sigma = \bigoplus_{a} F(aL \oplus w)$$

from eq.(3). Therefore, from eq.(4), eq.(5) and Lemma 4.1, we obtain that

$$\Pr_{L,w}(\sigma = (0, \dots, 0)) = \Pr_{L,w}(\bigoplus_{a} F(aL \oplus w) = (0, \dots, 0))
\ge \left(1/2 + 2^{2^{r+1}-1}(\mu - 1/2)^{2^{r+1}}\right)^{n} .$$
(6)

Let $\mu = 1 - \epsilon$, where ϵ is sufficiently small. Then the right hand side of eq.(6) can be approximated as

$$\left(1/2 + 2^{2^{r+1}-1} (\mu - 1/2)^{2^{r+1}} \right)^n \approx \left(1/2 + 1/2 (1 - 2 \times 2^{r+1} \epsilon) \right)^n$$

$$\approx 1 - 2^{r+1} n \epsilon$$

$$= p .$$

That is.

$$\Pr_{L,w}(\sigma = (0,\ldots,0)) \approx p .$$

Hence, in Algorithm 2, if k_c is the correct key,

$$\Pr_{L,w}(T_c = i) \approx \binom{N}{i} p^i (1-p)^{N-i} .$$

On the other hand, if k_w is a wrong key,

$$\Pr_{L,w}(T_w = j) = \binom{N}{j} 2^{-nj} (1 - 2^{-n})^{N-j}$$

because

$$\Pr_{L,w}(\sigma = (0, \dots, 0)) = 2^{-n} .$$

Consequently,

$$\Pr_{L,w}(T_c = i \text{ and } 0 \le T_w \le i - 1 \text{ for all } w \ne c)
= \Pr_{L,w}(T_c = i) \Pr_{L,w}(0 \le T_w \le i - 1 \text{ for all } w \ne c)
\approx {N \choose i} p^i (1-p)^{N-i} \left(\sum_{0 \le j \le i-1} {N \choose j} 2^{-nj} (1-2^{-n})^{N-j} \right)^{|K_m|-1} .$$

Since i ranges from 1 to N, the result follows.

Our experiment shows that if $N = \lceil p^{-2} \rceil$, then the success probability is larger than 90 %.

4.3 Example

We show a block cipher such that it is broken by the proposed attack, but not broken by the higher order differential attack.

 \mathcal{KN} cipher developed by Knudsen and Nyberg is provably secure against the differential attack and the linear attack [6]. It is a 6 round Feistel cipher such that n=32 and the round function G is given by

$$G(k,x) = d(f(e(x) \oplus k))$$
,

where $f(x) = x^3$ over GF(2³³), $d: \{0,1\}^{33} \to \{0,1\}^{32}$ discards one bit from its argument and

$$e(x_1,\ldots,x_{32})=(x_1,\ldots,x_{32},a_1x_1\oplus\cdots\oplus a_{32}x_{32})$$

for some a_1, \ldots, a_{32} . Since $\deg(G) = 2$, \mathcal{KN} cipher is broken with 512 chosen plaintext and 2^{41} complexity by the higher order differential attack [4].

Now consider a slight modification of KN cipher. Let the round function be

$$G'(k,x) = d(f(e'(x) \oplus k))$$
,

where

$$e'(x_1,\ldots,x_{32})=(x_1,\ldots,x_{32},x_1\cdots x_{32})$$
.

We call this cipher \mathcal{KN}' cipher. Then \mathcal{KN}' cipher cannot be broken by the higher order differential attack because $\deg(G') = 32$ which is very large.

However, it is broken by the proposed attack as follows. First, G' is $(2, 1 - 2^{-32})$ -expressible. Therefore, F of eq.(2) is $(2^3, (1 - 2^{-32})^3)$ -expressible. Now from Theorem 4.1, for N = 2, the last round key can be found with 2^{10} chosen plaintexts, 2^{42} complexity and the success probability almost 100%, where $p \approx 0.99$.

5 Higher Order Bent Function

In this section, we introduce a notion of higher order bent functions in order to prevent our attack. We then present their explicit constructions which satisfy some other cryptographic criteria as well.

5.1 Higher Order Nonlinearity

The truth table of a Boolean function f(x) is defined as $(f(\alpha_0), \ldots, f(\alpha_{2^n-1}))$, where α_i is a vector of length n representing i in binary. For two Boolean functions f(x) and g(x), let d(f(x), g(x)) denote the Hamming distance between $(f(\alpha_0), \ldots, f(\alpha_{2^n-1}))$ and $(g(\alpha_0), \ldots, g(\alpha_{2^n-1}))$.

Let $B^{(r)}(x)$ denote the set of Boolean functions with degree at most r for $0 \le r \le n$. That is,

$$B^{(r)}(x) = \{a_0 \oplus \bigoplus_{1 \le i \le n} a_i x_i \oplus \cdots \oplus \bigoplus_{1 \le i_1 < \cdots < i_r \le n} a_{i_1, \dots, i_r} x_{i_1} \cdots x_{i_r} \}.$$

Now we define the r-th order nonlinearity of a Boolean function f(x) as follows.

Definition 5.1. Let

$$N_f^{(r)} \stackrel{\triangle}{=} \min_{g(x) \in B^{(r)}(x)} d(f(x), g(x))$$

for $0 \le r \le n$. We say that $N_f^{(r)}$ is the r-th order nonlinearity of f(x).

Note that the well known nonlinearity of f(x) is equivalent to $N_f^{(1)}$.

We next show that $N_f^{(r)}$ is closely related to the covering radius of the r-th order Reed-Muller code.

Definition 5.2. [7] The r-th order Reed-Muller code $\mathcal{R}(r,n)$ of length 2^n , for $0 \le r \le n$, is the set of the truth table of a Boolean function f(x) such that $\deg(f) \le r$.

The covering radius of $\mathcal{R}(r,n)$ is defined as

$$\rho(r,n) \stackrel{\triangle}{=} \max_{v \in \{0,1\}^{2^n}} \min_{u \in \mathcal{R}(r,n)} d(v,u) .$$

Proposition 5.1. [2] If $0 \le r \le n-3$, then

$$\rho(r,n) \geq \begin{cases} 2^{n-r-3}(r+4) & \text{if r is even} \\ 2^{n-r-3}(r+5) & \text{if r is odd} \end{cases}.$$

Theorem 5.1.

$$\max_{f(x)} N_f^{(r)} = \rho(r, n) .$$

Proof. From the definition of the r-th order nonlinearity $N_f^{(r)}$,

$$\max_{f(x)} N_f^{(r)} = \max_{f(x)} \min_{g(x) \in B^{(r)}(x)} d(f(x), g(x)) .$$

Since $B^{(r)} = \{u \mid u \in \mathcal{R}(r, n)\},$ we have

$$\max_{f(x)} \min_{g(x) \in B^{(r)}(x)} d(f(x), g(x)) = \max_{v \in \{0,1\}^{2^n}} \min_{u \in \mathcal{R}(r,n)} d(v, u)$$
$$= \rho(r, n) .$$

5.2 Higher Order Bent Function

We then define r-th order bent functions based on Theorem 5.1 and Proposition 5.1 as follows.

Definition 5.3. We say that f(x) is an r-th order bent function if

$$N_f^{(r)} \geq \begin{cases} 2^{n-r-3}(r+4) & \text{if r is even} \\ 2^{n-r-3}(r+5) & \text{if r is odd} \end{cases}.$$

for $0 \le r \le n-3$.

(A well known bent function is also a 1-st order bent function. However, the converse is not true.)

5.3 Basic Construction

In what follows, let $x = (x_1, \ldots, x_n)$ and $x' = (x_1, \ldots, x_{n-1})$. For a Boolean function f(x), let

$$f_1(x') \stackrel{\triangle}{=} f(x',0)$$
 and $f_2(x') \stackrel{\triangle}{=} f(x',1)$.

Lemma 5.1.

$$N_f^{(r)} \ge N_{f_1}^{(r)} + N_{f_2}^{(r)}$$
.

Proof.

$$\begin{split} N_f^{(r)} &= \min_{g(x) \in B^{(r)}(x)} d(f(x), g(x)) \\ &= \min_{g(x) \in B^{(r)}(x)} d(f(x', 0), g(x', 0)) + d(f(x', 1), g(x', 1)) \\ &\geq \min_{g_1(x') \in B^{(r)}(x')} d(f_1(x'), g_1(x')) + \min_{g_2(x') \in B^{(r)}(x')} d(f_2(x'), g_2(x')) \\ &= N_{f_1}^{(r)} + N_{f_2}^{(r)} \ . \end{split}$$

Lemma 5.2. If $f_1(x') = f_2(x')$, then

$$N_f^{(r)} = 2N_{f_1}^{(r)}$$
 .

Proof. First $N_f^{(r)} \ge 2N_{f_1}^{(r)}$ from Lemma 5.1. Next choose $g'(x') \in B^{(r)}(x')$ such that

$$d(f_1(x'), g'(x')) = N_{f_1}^{(r)}$$

arbitrarily. Define g(x) as g(x) = g'(x'). Then

$$\begin{split} 2N_{f_1}^{(r)} &= N_{f_1}^{(r)} + N_{f_2}^{(r)} \\ &= d(f_1(x'), g'(x')) + d(f_2(x'), g'(x')) \\ &= d(f(x', 0), g(x', 0)) + d(f(x', 1), g(x', 1)) \\ &= d(f(x), g(x)) \\ &\geq \min_{g(x) \in B^{(r)}(x)} d(f(x), g(x)) \\ &= N_{\varepsilon}^{(r)} \end{split}$$

because $g(x) \in B^{(r)}(x)$. Therefore, $N_f^{(r)} = 2N_{f'}^{(r)}$.

Let

$$\sigma^{(r)}(x) = \bigoplus_{1 \le i_1 < \dots < i_r \le n} x_{i_1} \cdots x_{i_r} .$$

for $0 \le r \le n$. Then McLoughlin showed a lower bound on $\rho(n-3,3)$ by using $\sigma^{(r)}(x)$ [9]. It can be restated as follows.

Proposition 5.2. $\sigma^{(n-2)}(x)$ is an (n-3)-th order bent function for $n \geq 3$.

Now we show our basic construction of r-th order bent functions.

Theorem 5.2. Let

$$f(x_1,\ldots,x_n) = \sigma^{(r+1)}(x_1,\ldots,x_{r+3})$$
.

Then f(x) is an r-th order bent function for $0 \le r \le n-3$.

Proof. By using Lemma 5.2 repeatedly n-r-3 times, we have

$$N_f^{(r)} = 2^{n-r-3} N_{\sigma^{(r+1)}}^{(r)}$$
.

Then from Proposition 5.2, we see that

$$N_f^{(r)} \ge \begin{cases} 2^{n-r-3}(r+4) & \text{if } r \text{ is even }, \\ 2^{n-r-3}(r+5) & \text{if } r \text{ is odd }. \end{cases}$$

5.4 Improved Construction (I)

The r-th order bent function obtained from Theorem 5.2 is cryptographically weak since it is not balanced and x_{r+4}, \ldots, x_n do not appear in f(x). In what follows, we show some improved constructions.

Definition 5.4. A Boolean function f(x) is balanced if

$$|\{x \mid f(x) = 0\}| = |\{x \mid f(x) = 1\}|$$
.

Definition 5.5. A Boolean function f(x) satisfies SAC if

$$f(x) \oplus f(x \oplus \alpha)$$

is balanced for any α such that the Hamming weight of α is equal to 1.

Lemma 5.3. If deg(f(x)) > r and $deg(h(x)) \le r$, then

$$N_f^{(r)} = N_{f \oplus h}^{(r)} .$$

Proof.

$$\begin{split} N_{f \oplus h}^{(r)} &= \min_{g(x) \in B^{(r)}(x)} d(f(x) \oplus h(x), g(x)) \\ &= \min_{g(x) \in B^{(r)}(x)} d(f(x), g(x) \oplus h(x)) \\ &= \min_{g(x) \in B^{(r)}(x)} d(f(x), g(x)) \\ &= N_f^{(r)} \ . \end{split}$$

By using Lemma 5.3, we can prove the following theorems. The proofs will be given in the final paper.

Theorem 5.3. Suppose that r + 3 < n. Let

$$f(x_1,\ldots,x_n)=\sigma^{(r+1)}(x_1,\ldots,x_{r+3})\oplus x_{r+4}\oplus\cdots\oplus x_n.$$

Then f(x) is a balanced r-th order bent function.

Theorem 5.4. Suppose $2 \le r \le n-3$. Let

$$f(x_1,\ldots,x_n)=\sigma^{(r+1)}(x_1,\ldots,x_{r+3})\oplus(x_1\oplus\cdots\oplus x_{r+3})(x_{r+4}\oplus\cdots\oplus x_n).$$

Then f(x) is an r-th order bent function which satisfies SAC.

Theorem 5.5. There exist r-th order bent functions which satisfy PC(l) of order k.

5.5 Improved Construction (II)

Next we show r-th order bent functions such that each x_i is involved in a large degree term.

Lemma 5.4. For any r, let

$$\begin{cases} s_n(x_1,\ldots,x_n) \stackrel{\triangle}{=} \sigma^{(r+1)}(x_1,\ldots,x_n) , \\ s_{n-1}(x_1,\ldots,x_{n-1}) \stackrel{\triangle}{=} \sigma^{(r+1)}(x_1,\ldots,x_{n-1}) . \end{cases}$$

Then,

$$N_{s_n}^{(r)} \ge 2N_{s_{n-1}}^{(r)}$$
.

Proof. Note that

$$\sigma^{(r+1)}(x',0) = \sigma^{(r+1)}(x')$$

$$= s_{n-1}(x') ,$$

$$\sigma^{(r+1)}(x',1) = \sigma^{(r+1)}(x') \oplus \sigma^{(r)}(x')$$

$$= s_{n-1}(x') \oplus \sigma^{(r)}(x') .$$

Then from Lemma 5.1 and Lemma 5.3,

$$\begin{split} N_{s_n}^{(r)} &\geq N_{s_{n-1}}^{(r)} + N_{s_{n-1}}^{(r)} \\ &= 2N_{s_{n-1}}^{(r)} \ . \end{split}$$

Theorem 5.6. Let

$$f(x_1,\ldots,x_n) \stackrel{\triangle}{=} \sigma^{(r+1)}(x_1,\ldots,x_n)$$
.

Then f(x) is an r-th order bent function for $0 \le r \le n-3$.

Proof. Let

$$s_{r+3}(x_1,\ldots,x_{r+3}) \stackrel{\triangle}{=} \sigma^{(r+1)}(x_1,\ldots,x_{r+3})$$
.

Then by using Lemma 5.4 repeatedly n-r-3 times, we have

$$N_f^{(r)} \ge 2^{n-r-3} N_{s_{r+3}}^{(r)}$$
.

Finally from Proposition 5.2, we see that

$$N_f^{(r)} \ge \begin{cases} 2^{n-r-3}(r+4) & \text{if } r \text{ is even }, \\ 2^{n-r-3}(r+5) & \text{if } r \text{ is odd }. \end{cases}$$

Therefore, f(x) is an r-th order bent function.

Note that each x_i is involved in a term of degree (r+1) in the above f.

References

- E.Biham and A.Shamir. Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag, 1993.
- G.D.Cohen, M.G.Karpovsky, H.F.Mattson, Jr. and J.R.Schatz. Covering Radius Survey and Recent Results. In *IEEE Transactions on Information Theory*, volume 31, Number 3, pages 328–343, 1985. 69
- T.Jakobsen. Cryptanalysis of block ciphers with probabilistic non-linear relations of low degree. In Advances in Cryptology — CRYPTO' 98 Proceedings, volume 1462 of Lecture Notes in Computer Science, pages 212–222, Springer-Verlag, 1998.
- T.Jakobsen and L.R.Knudsen. The interpolation attack on block ciphers. In Fast Software Encryption, volume 1267 of Lecture Notes in Computer Science, pages 28–40, Springer-Verlag, January 1997. 62, 64, 68

- X. Lai. Higher order derivatives and differential cryptanalysis. In Proceedings of Symposium on Communication, Coding and Cryptography, in honor of James L. Massey on the occasion of his 60'th birthday, February 10–13, 1994, Monte-Verita, Ascona, Switzerland, 1994. 64
- K.Nyberg and L.R.Knudsen. Provable security against a differential attack. In Journal of Cryptology, volume 8, number 1, pages 27–37, Winter 1995.
- F.J.MacWilliams and N.J.A.Sloane. The theory of error-correcting codes. North-Holland, 1977. 69
- M.Matsui. Linear cryptanalysis method for DES cipher. In Advances in Cryptology EUROCRYPT' 93 Proceedings, volume 765 of Lecture Notes in Computer Science, pages 386–397, Springer-Verlag, 1993.
- 9. A.M.McLoughlin. The covering radius of the (m-3)-rd order Reed-Muller codes and lower bounds on the (m-4)-th order Reed-Muller codes. In SIAM Journal of Applied Mathematics, volume 37, number 2, October 1979. 71
- 10. J.Pieprzyk and G.Finkelstein. Towards effective nonlinear cryptosystem design. In *IEE Proceedings Part E*, volume 35, number 6, pages 325–335, November 1988.

Fast Algorithms for Elliptic Curve Cryptosystems over Binary Finite Field

Yongfei Han¹, Peng-Chor Leong², Peng-Chong Tan², and Jiang Zhang¹

¹ APDC Security & Crypto Dept Gemplus Corporate R & D 89, Science Park Drive #04-01/05 Singapore 118261 yfh69@hotmail.com ² Centre for Advanced Information Systems, School of Applied Science, Nanyang Technological University, Singapore 639798

Abstract. In the underlying finite field arithmetic of an elliptic curve cryptosystem, field multiplication is the next computational costly operation other than field inversion. We present two novel algorithms for efficient implementation of field multiplication and modular reduction used frequently in an elliptic curve cryptosystem defined over $GF(2^n)$. We provide a complexity study of the two algorithms and present an implementation performance of the algorithms over $GF(2^{167})$.

Keywords: Galois field arithmetic, Elliptic Curve Cryptosystems, field multiplication, modular reduction.

1 Introduction

In 1985, Neil Koblitz and Victor Miller independently proposed the elliptic curve cryptosystem, whose security rests on the discrete logarithm problem over points on an elliptic curve. Elliptic curve cryptography can be used to provide both a digital signature scheme and an encryption scheme. With the apparent advantage of high cryptographic strength relative to key size, elliptic curve cryptosystems [9,14] have gained much popularity in the implementation of discrete logarithm based public key protocols. The shorter key size generally leads to improved computational efficiencies and smaller storage and bandwidth requirements. Although elliptic curve cryptosystem can be based on finite field of any characteristic, it is generally practical to implement within the prime or binary finite field [9,14].

Certain classes of elliptic curves such as the subfield curves, supersingular and anomalous binary curves have been proposed which provide improved efficiencies in implementation. However the extra structure provided by these curves are subjected to attack and reviewed recently [17,4]. We consider only non-supersingular and non-anomalous elliptic curves over non-composite field in the paper. The algorithms presented are specifically for binary finite field with standard basis representation.

Efficient implementation of elliptic curve cryptography can be focused on 2 levels. At elliptic curve group level, fast algorithms for multiplying a base point P of an elliptic curve may be applied [5,6,7]. The computation of multiplying a point P of an elliptic curve group by a large integer d is analogous to exponentiation of an element in a multiplicative group to the d^{th} power. The generally accepted algorithm for the computation is the "square-and-multiply" algorithm. Signed digit representation, k-SR representation, addition chains and sliding window methods are applied to the computation of scalar multiplication, as the are employed to exponentiation [1,2].

For the underlying finite field arithmetic, more efficient algorithms that speeds up computation of field multiplication and inversion may be introduced. Field multiplication is the next costly operation other than field inversion. Various algorithms, such as the transformation to projective coordinates trade field inversion for field multiplication. Hence, it is desirable to provide fast and effective field multiplication and modular reduction.

The purpose of this paper is to present new approaches for field multiplication and a modular reduction commonly performed in elliptic curve cryptosystems defined over binary finite field.

First, we review previous works in section 2. In section 3, we present a method to speed up computation of field multiplication. This algorithm is applicable to standard basis representation of elements in Galois field $GF(2^n)$. The algorithm is based on modified classical "shift-and-add" method. Through elimination of extensive shiftings, our algorithm is suited for microprocessors that have small word size, and only instruction that can shift only one bit at a time. Such microprocessors are common in 8 or 16 bit microcontrollers and smartcards. While there exist fast binary finite field multiplication with the use of table look-up [10], such algorithms are generally not suitable for computing multiplication of field elements with degree > 5 using low end microprocessor with limited memory.

In section 4, we present an efficient modular reduction based on optimization of Schroeppel's modular reduction technique [16], and our method is more efficient than Schroeppe;'s approach. Detailed analysis of the complexity and performances of our algorithms will also be presented.

We conclude this article with the comparison of implementation results for field multiplication and reduction over $GF(2^{167})$. Relative to the the classical "shift-and-add" method of multiplication, our implementation result shows approximately 12 percent reduction in computation time for a general purpose 32 bits microprocessor. As for the field modular reduction, 14 percent reduction of the computation cost can be realised.

2 Previous Works

An elliptic curve, defined on a field $K = GF(2^n)$ where n is a prime, is the set of solution points(x, y) to an equation of the form:

$$y^2 + xy = x^3 + ax + b$$

with $a, b \in K$.

The set of points on an elliptic curve, together with a special point called the *point of infinity* can be equipped with an Abelian group structure by the following point addition operation:

$$\lambda = \frac{y_1 + y_2}{x_1 + x_2}$$

$$x = a + \lambda^2 + \lambda + x_1 + x_2$$

$$y = (x_1 + x)\lambda + x + y_1$$

And by following point doubling operation:

$$\lambda = x_1 + \frac{y_1}{x_1}$$

$$x_2 = a + \lambda^2 + \lambda$$

$$y_2 = x_1^2 + \lambda x_2 + x_2$$

The simplest technique to compute multiplication in GF(2) is to use the "shift-and-add" method. As no arithmetic carry over is involved, the "shift-and-add" method is a neat and easy method for implementation. Addition in GF(2) is simply the bitwise exclusive-or operation.

Selection of an elliptic curve is a critical step before the implementation. The curve selected should not be a supersingular curve or anomalous curve.

For computation of field multiplication over $GF(2^n)$, we noted that word level multiplication in GF(2) is usually not supported in general microprocessors. There are 2 common software implementation techniques to achieve the GF(2) multiplication.

- Table look-up method
- Emulation using "shift-and-add" technique

In Table look-up method [10], the field multiplication result are first precomputed. A simple method is to use 2 tables, to store the higher order and lower order of the multiplication result. The tables are addressed using the bits of the multiplier and multiplicand. Therefore a 8-bits word for GF(2) multiplication would require $2 \times 2^8 \times 2^8 \times 8$ bits = 128 Kbyte of storage space for the look up tables. For 16 bit operands, $2 \times 2^{16} \times 2^{16} \times 16$ bits = 16 GByte of look up table would be required. Although there exists techniques [10] to handle 16 bit GF(2) multiplication using 8-bits look up table, we noted that the overheads is not favourable for microprocessors without special shift instruction and not practical with devices with extremely limited memory.

The simplest technique to compute multiplication in GF(2) is to use the "shift-and-add" method. Addition in GF(2) is simply the bitwise exclusive-or operation. As no arithmetic carry over is involved, the "shift-and-add" method is a neat and simple method for implementation.

3 A New Approach for Multiplication

To compute the multiplication of two field elements A and B in standard basis over $GF(2^n)$, the classical "shift-and-add" algorithm as described in [16] is commonly used. The classical method typically incurs computational cost of shifting 2s(n-1) bits of the intermediate results; where n is the number of bits of the field.

Our algorithm attempts to eliminate the extensive number of shift operations which inherently contributes to a large part of the computational cost. Our method requires shifting 2s(w-1) bits of the intermediate results; where w is the wordsize of the microprocessor, and $s = \lceil n/w \rceil$. This contributes to greater performance improvement, particularly for microprocessors with small word size. It is noted that the number of field additions remains the same as classical method since it is dependent on the hamming weight of the multiplier. As 'addition' in GF(2) operation does not involve 'carry', with the addition operator defined as exclusive-or operation, the saving in shift operations is possible with our new algorithm.

3.1 Efficient Field Multiplication Algorithm

Definition 1. Let $A, B = (b_{sw-1}...b_1b_0) \in GF(2^n)$ be the bit-string representation of the multiplier B. B can be partitioned into s blocks and each block is of length w bits, and $s = \lceil n/w \rceil$. Denote $t_i = A \cdot b_i \cdot x^i, i \in \{0..n-1\}$

The field multiplication result of $C = A \times B = A \sum_{i=0}^{w-1} x^i \left(\sum_{j=0}^{s-1} \left(b_{jw+i} x^{jw} \right) \right)$ where $B = (b_{sw-1}...b_1b_0) \in GF(2^n)$ can be re-expressed as:

$$C = (t_0 + t_w x^w + t_{2w} x^{2w} + \dots + t_{(s-1)w}) x^{(s-1)w} + (t_1 x + t_{w+1} x^{w+1} + t_{w+1} x^{2w+1} + \dots + t_{(s-1)w+1} x^{(s-1)w+1})$$

$$+ \dots + (t_{w-1} x^{w-1} + t_{w+(w-1)} x^{w+(w-1)} + t_{2w+(w-1)} x^{2w+(w-1)}$$

$$+ \dots + t_{(s-1)w+(w-1)} x^{(s-1)w+(w-1)})$$

$$= A(b_0 + b_w . x^w + b_{2w} . x^{2w} + \dots + b_{(s-1)w} . x^{(s-1)w})$$

$$+ A(b_1 . x^1 + b_{w+1} . x^{w+1} + b_{2w+1} . x^{2w+1} + \dots + b_{(s-1)w+1} . x^{(s-1)w+1})$$

$$+ \dots + A(b_{w-1} . x^{w-1} + b_{2w-1} . x^{2w-1} + b_{3w-1} . x^{3w-1} + \dots + b_{sw-1} . x^{sw-1})$$

$$= A(b_0 + b_w . x^w + b_{2w} . x^{2w} + \dots + b_{(s-1)w} . x^{(s-1)w})$$

$$+ x(A(b_1 + b_{w+1} . x^w + b_{2w+1} . x^{2w} + \dots + b_{(s-1)w+1} . x^{(s-1)w})$$

$$+ \dots + x^{w-1} (A(b_{w-1} + b_{w+(w-1)} . x^w + b_{2w+(w-1)} . x^{2w} + \dots + b_{(s-1)w+(w-1)} . x^{(s-1)w}) \dots)$$

The efficient field multiplication is based on the following model of computing the intermediate results of $(t_0, t_w, ..., t_{(s-1)w})$ first and progressively on the next tuple $(t_1, t_{w+1}, ..., t_{(s-1)w+1})$, until all the intermediate results are computed in which the last tuple is $(t_{w-1}, t_{w+(w-1)}, ..., t_{(s-1)w+(w-1)})$.

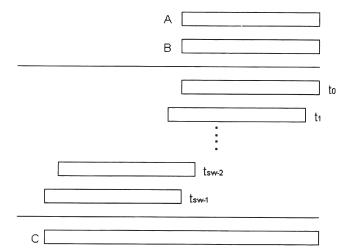


Fig. 1. Model of improved multiplication algorithm

Algorithm 1 Denote LeftShift(X) as left shifting the coefficient representation of the polynomial X by 1 bit and C[j] denotes the j^{th} word of the coefficient representation of polynomial C where $j \in \{0..(s-1)\}$.

```
NEW-METHOD-FOR-FIELD MULTIPLICATION (A, B)
      C \leftarrow 0
  2
      for j \leftarrow w - 1 down to 0
          do p \leftarrow 0
  3
              for k \leftarrow 1 down to s
  4
  5
                  do if B_{kw-1} = 1
  6
                          then for i \leftarrow (s-1) + p down to p
                                     do C[i] \leftarrow C[i] \oplus A[i-p]
  7
  8
                       p \leftarrow p + 1
  9
              LeftShift(C)
              LeftShift(B)
 10
 11
      return C
```

	No. of shift operations		
Shift operation cost	"Shift-and-add" method	Our method	
Number of shift operation on C	2s(n-1)	2s(w-1)	
Number of shift operation on B	s(w-1)	s(w-1)	
Total shift operations	2s(n-1) + s(w-1)	2s(w-1) + s(w-1)	

Table 1. Computational time cost comparison for field multiplication

Using the classical "shift-and-add" method, 2s(n-1) shift operations are required to compute the field multiplication in $GF(2^n)$. With the new approach, only 2s(w-1) shift operations are incurred without any need for precomputation. The relation between the number of bits n of the underlying field and the word size, w would determine if the new algorithm would be more efficient compared to the binary method. The new algorithm will perform even more efficiently than the binary method when

$$2s(w-1) < 2s(n-1)$$
 or equivalently $w < n$

The word size of general microprocessor are usually 8, 16, 32 and 64 bits and for elliptic curve over $GF(2^n)$, n is usually chosen to be about 160 bits. It is noted that when the field size of the elliptic curve is increased, the new algorithm will perform more efficiently compare to the classical "shift-and-add" field multiplication method. This is because the number of shift operation performed on element C would remain unchanged for the new approach, whereas the number of shift operations using the "shift-and-add" method would depend on the field size n.

Table 2 compares the two methods based on a typical 167 bits field for elliptic curve cryptosystem, with s defined as the number of words, and w as the wordsize of the microprocessor.

16	No. of shift operations			
w	s	"Shift-and-add" method	Our method	Percentage Savings
8	21	7119	441	94%
16	11	3817	495	87%
32	6	2178	558	74%
64	3	1185	567	52%

Table 2. Computational time cost comparison for $GF(2^{167})$ field multiplication

4 Modular Reduction

The result of field multiplication requires storage length of $2 \times sw$. Modular reduction can be done very efficiently with an irreducible polynomial, such as trinomial and pentanomial using shifts and additions. The idea is to zero out the upper bits and add the representation of each original term right shifted by some quantity. Schroeppel et al. describes a practical approach of working on one computer word at a time to systematically perform the polynomial modular reduction [16].

We consider a trinomial modulus of the expression $x^n + x^k + 1$, where n = 167, k = 6. After each field multiplication or squaring, the result must be reduced modulo $F(x) = x^{167} + x^6 + 1$.

The product of two polynomials of degree 166 produces a polynomial of degree 332. Assume the polynomial to be reduced is:

$$P(x) = a_{332}x^{332} + \dots + a_1x + a_0$$

Then the reduction modulo $x^{167}+x^6+1$ proceeds by reducing each term modulo the trinomial and subtracting it from the result. We noted that:

$$x^{167} \equiv x^6 + 1$$

$$x^n \equiv x^{n-161} + x^{n-167} \pmod{F(x)}$$

Instead of working on one computer word at a time, and lowering the degree of the polynomial by a word, proceeding from the high order terms to the low, our approach is to work on $\frac{s}{2}$ words at a time and lowering the degree by $\frac{s}{2}$. For our approach to be effective, it is therefore desirable to choose a trinomial with low k degree.

Algorithm 2 Let A be the result of field multiplication or squaring prior to modular reduction. A has degree of at most 2n-2. A can be partitioned into 2s blocks and each block is of length w bits. Let A_i denotes the i^{th} block of the partition of field element A. CarryRightShift(Q,d,T) denotes right shifting the memory location range in Q by d bits making use of the word shift with carry instruction available in general microprocessor, and that the carry bits are stored in T. Temp1 and Temp2 are registers of wordsize w. The following algorithm performs the modular reduction on A using the trinomial modulus, $x^n \equiv x^k + 1 \pmod{x^n + x^k + 1}$, when $k < \lfloor \frac{n}{2} \rfloor$ and n - k > w.

```
New-method-for-trinomial-modular-reduction (A)
   \begin{array}{ll} 1 & p \leftarrow ((n-k) \bmod w) \\ 2 & q \leftarrow \left\lfloor \frac{n-k}{w} \right\rfloor \\ 3 & u \leftarrow (n \bmod w) \end{array}
   4 \quad t \leftarrow \left| \frac{n}{w} \right|
   5 Temp1 \leftarrow 0
   6 Temp2 \leftarrow 0
       CarryShiftRight (A_{2s-1}...A_{2s-\lceil \frac{s}{2} \rceil}, p, Temp1)
         for j \leftarrow 2s - 1 down to 2s - \left\lceil \frac{s}{2} \right\rceil
   8
   9
       do A_{i-t} \leftarrow A_{i-t} \oplus A_i
         A_{2s-\left\lceil\frac{s}{2}\right\rceil-t-1} \leftarrow A_{2s-\left\lceil\frac{s}{2}\right\rceil-t-1} \oplus Temp1
 10
          CarryShiftRight (A_{2s-1}...A_{2s-\lceil \frac{s}{2} \rceil}, u-p, Temp1)
 11
         for j \leftarrow 2s - 1 down to 2s - \left\lceil \frac{s}{2} \right\rceil
 12
 13
           do A_{j-q} \leftarrow A_{j-q} \oplus A_j
         A_{2s-\left\lceil\frac{s}{2}\right\rceil-q-1} \leftarrow A_{2s-\left\lceil\frac{s}{2}\right\rceil-q-1} \oplus Temp1
 14
          CarryShiftRight (A_{s-1+\left|\frac{s}{2}\right|}...A_{s}, p, Temp1)
 15
         for j \leftarrow s - 1 + \left| \frac{s}{2} \right| down to s
 16
 17
           do A_{j-t} \leftarrow A_{j-t} \oplus A_j
 18
         A_{s-t-1} \leftarrow A_{s-t-1} \oplus Temp1
          CarryShiftRight (A_{s-1+\left|\frac{s}{2}\right|}...A_{s}, u-p, Temp1)
         for j \leftarrow s - 1 + \left| \frac{s}{2} \right| down to s
 20
          do A_{i-q} \leftarrow \bar{A_{j-q}} \oplus A_j
 21
         A_{s-q-1} \leftarrow A_{s-q-1} \oplus Temp1
mask \leftarrow (2^w - 1) \ll (n \bmod w)
 22
 23
         Temp1 \leftarrow A_{s-1} \wedge mask
 24
          CarryShiftRight (Temp1, p, Temp2)
 25
         A_{(s-1)-q} \leftarrow A_{(s-1)-q} \oplus Temp1
 26
         A_{(s-1)-q-1} \leftarrow A_{(s-1)-q-1} \oplus Temp2
         return C \leftarrow (A_{(s-1)}...A_0)
```

Table 3 compares the computational cost on the number of shift and addition operations required for Schroeppel's method and our improved method.

	No. of operations	
Operation cost	Schroeppel's method	Improved method
No. of shift operations on A	s(p + (w - p))	s(max(p, u))
No. of shift operations on		
temporary variables	p + (w - p)	$3 \times \max(p, u)$
Total shift operations	$(s+1) \times w$	$(s+3)\times(\max(p,u))$
Total no. of Field Additions		2s + 6

Table 3. Computational cost comparison for field modular reduction

A careful choice of reduction trinomial that has small value of $p = ((n - k) \mod w)$ and $u = (n \mod w)$ will boost efficiencies in our new algorithm. Further to the elimination of extensive shifting, the number of field additions is also reduced by a factor of 2 in our approach, this is achieved with the alignment on the degrees of the congruent terms with the microprocessor word size.

5 Performance of Implementation

The following table present the performance benchmark of the improved field multiplication and modular reduction in an elliptic curve cryptosystem defined over $GF(2^{167})$. The computation is based on C source codes compiled with Microsoft Visual C++ 5.0 without compiler's optimization. An Intel Pentium II 32 bit microprocessor running at 333 MHz was used to conduct the benchmarking.

Field Arithme	tic Classical me	ethod Our method	od Percentage Savings
Multiplication	n 0.27ms	$0.20 \mathrm{ms}$	12%
Modular Reduc	tion $0.07 \mathrm{ms}$	$0.06 \mathrm{ms}$	14%

Table 4. Computational timing cost for field multiplication and modular reduction

Comparing our new approaches of field multiplication and modular reduction to the classical methods, the timing results shows about 12 percent improvement for the multiplication, and approximately 14 percent improvement is achieved for the modular reduction.

References

- Yongfei Han, C. Mitchell, D. Gollmann, "Minimal Weight k-SR Representation." In Proceedings of fifth IMA Conference on Cryptography and Coding, LNCS1025, pages 34-43, Circnester, U. K, 1995. Springer-Verlag, Berlin. 76
- D. Gollmann, Yongfei Han, C. Mitchell, "Redundant integer representations and fast exponentiation." Designs, Codes and Cryptography, 7 (1996), pages 135-151.
 76
- 3. Erik Dewin, "Fast software Implementation for Arithmetic Operations in $GF(2^n)$ ", Advances in Cryptology, Proceedings Asiacrypt '96, LNCS 1163, Springer-Verlag, 1996, p.p 65-76
- 4. Robert Gallant, Robert Lambert, Scott Vanstone, Improving the Parallelized Pollard Lambda Search on Binary Anomalous Curves, P1363 Standards Internet Web Site, 1998. Online. Available: http://grouper.ieee.org/groups/1363/contributions 75

- Jorge Guajardo, Christof Paar, "Efficient Algorithms for Elliptic Curve Cryptosystem", CRYPTO '97, Springer-Verlag, LNCS 1294, pp. 342-356, 1997 76
- Yongfei Han, Jiang Zhang, Peng-Chong Tan, "Efficient Elliptic Curve Cryptosystems", International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99), Hong Kong, 1999.
- Yongfei. Han, J. Zhang, P.-C. Tan," Direct Computation for Elliptic Curve Cryptosystem", Workshop on Cryptographic Hardware and Embedded Systems, Worcester Polytechnic Institute, Worcester, Massachusetts, 1999.
- 8. D.E Knuth, The Art of Computer Programming, Vol.2: Seminumerical Algorithms, 2nd edition, Addison-Wesley, 1981.
- N. Koblitz, "Elliptic curve cryptosystems," Mathematics of Computation 48 (1987), 203-209
- 10. Cetin K. Koc, Tolga Acar. "Montgomery Multiplication in $GF(2^k)$ ", Design, Codes and Cryptography, 1-14(1997), Kluwer Academic Publishers, Boston, 1997 76, 78
- R. Lidl and H. Niederreiter. Introduction to Finite Fields and Their Applications. Cambridge University Press, Cambridge, UK, 1994
- 12. A.J Menezes Elliptic Curve Public Key Cryptosystems. Kluwer Academic Publishers, Boston, MA, 1993
- R. J. McEliece. Finite Fields for Computer Scientist and Engineers. Kluwer Academic Publishers, 1987
- V. Miller. Uses of elliptic curves in cryptography, In Advances in Cryptology -CRYPTO '85, pages 417-426. Springer-Verlag, Berlin 1986. 75
- 15. Francois Morain et J. Olivos, "Speeding up the Computation on an Elliptic Curve using Addition-Subtraction Chains", RAIRO Informatique Theorique Et Applications Theoretical Informatics and Applications, 24,6, p. 531-543, 1990.
- R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck. "Fast key exchange with elliptic curve systems". In D. Coppersmith, editor, Advances in Cryptology CRYPTO 95, Lecture Notes in Computer Science, No. 973, pages 43-56, New York, NY, 1995. Springer-Verlag. 76, 78, 82
- 17. Michael J. Wiener, Robert J, Zuccherato, Faster Attacks on Elliptic Curve Crytosystems, IEEE P1363 Standards Internet Web Site, April 1998. Online. Available: http://grouper.ieee.org/groups/1363/contributions 75

Optimizing the Menezes-Okamoto-Vanstone (MOV) Algorithm for Non-Supersingular Elliptic Curves

Junji Shikata¹, Yuliang Zheng², Joe Suzuki¹, and Hideki Imai³

Department of Mathematics, Graduate School of Science, Osaka University, 1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan,

{shikata, suzuki}@math.sci.osaka-u.ac.jp,

² School of Comp. and Info. Tech., Monash University, McMahons Road, Frankston, Melbourne, Victoria 3199, Australia,

yuliang@pscit.monash.edu.au,

³ Institute of Industrial Science, University of Tokyo, 7-22-1 Roppongi, Minato-ku, Tokyo 106-8558, Japan, imai@iis.u-tokyo.ac.jp

Abstract. We address the Menezes-Okamoto-Vanstone (MOV) algorithm for attacking elliptic curve cryptosystems which is completed in subexponential time for supersingular elliptic curves. There exist two hurdles to clear, from an algorithmic point of view, in applying the MOV reduction to general elliptic curves: the problem of explicitly determining the minimum extension degree k such that $E[n] \subset E(F_{a^k})$ and that of efficiently finding an n-torsion point needed to evaluate the Weil pairing, where n is the order of a cyclic group of the elliptic curve discrete logarithm problem. We can find an answer to the first problem in a recent paper by Balasubramanian and Koblitz. On the other hand, the second problem is important as well, since the reduction might require exponential time even for small k. In this paper, we actually construct a novel method of efficiently finding an n-torsion point, which leads to a solution of the second problem. In addition, our contribution!! allows us to draw the conclusion that the MOV reduction is indeed as powerful as the Frey-Rück reduction under n/q-1, not only from the viewpoint of the minimum extension degree but also from that of the effectiveness of algorithms.

1 Introduction

1.1 History and Motivation

In 1985, Koblitz [14] and Miller [20] independently proposed the use of elliptic curves over finite fields for public-key cryptography. Since that time, elliptic curve cryptosystems have gained a tremendous amount of attention and many researchers have devoted their time to the study of elliptic curves.

The security of elliptic curve cryptosystems is based on the presumed intractability of the *Elliptic Curve Discrete Logarithm Problem*, which we abbreviate as the ECDLP. More specifically, the ECDLP can be stated as follows: Let E be an elliptic curve defined by

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, (a_1, a_2, a_3, a_4, a_6 \in F_q)$$

where F_q is a finite field with $q = p^m$ (p: a prime number) elements. Given a base point $P \in E(F_q)$ and $R \in \langle P \rangle$, one is asked to find an integer l such that R = lP, where $E(F_q)$ is the set of its F_q -rational points.

In general, thus far, it is believed that the ECDLP requires exponential time in $\log q$ to solve. Nevertheless, it has been known that, for some special cases, the ECDLP is no more difficult than the Discrete Logarithm Problem (DLP) in finite fields. Significant developments in this line of research are represented by the Menezes-Okamoto-Vanstone (MOV) algorithm [19], the Frey-Rück (FR) algorithm [10] and the Semaev-Smart-Satoh-Araki (SSSA) algorithm [26][29][23].

In the following discussion, we assume that $n = \#\langle P \rangle$, the order of a base point P, is a prime number. This condition is not restrictive, since we can reduce the composite case to the prime one by applying the Chinese Remainder Theorem and the Pohlig-Hellman algorithm.

Technically, the SSSA algorithm reduces the ECDLP to the DLP of the additive group structure of the base field for so-called anomalous curves and solves it in polynomial time. (For more details, see [26][29][23].) Thus, in the sequel, we will also assume that n/q, since the SSSA algorithm can be applied to the case of n|q.

In contrast, the MOV and FR algorithms reduce the ECDLP with the above assumptions (n prime and n/q) to the DLP in the multiplicative subgroup of an extension field F_{q^k} of the base field F_q and then solve the DLP using the currently known best algorithm. (For example, see [8].) A natural question that arises from an algorithmic point of view is whether it is possible to realize the reductions (i.e. transformations from the ECDLP to the DLP in finite fields) in such a way that they work efficiently.

For the FR reduction, the above question has already been answered (For example, see [11][12]): it is known that the FR reduction can work in probabilistic polynomial time in $k \log q$. Here k is explicitly given as the smallest positive integer with $q^k \equiv 1 \mod n$. (Note that this condition follows from the requirement that F_{q^k} must contain n-th roots of unity.) Thus, if such a k is small enough to solve the DLP in $F_{q^k}^*$ in subexponential time in $\log q$, the reduction itself is always completed in polynomial time in $\log q$. Consequently, in such a case, the FR algorithm is completed in subexponential time in $\log q$. In particular, if n|q-1, we have no need to extend the base field F_q and the FR reduction can be easily applied.

On the other hand, for the MOV reduction, the above question has not been explicitly answered yet: thus far, it is well known that, for supersingular curves,

1. the necessary minimum extension degree k is at most six; and

2. the MOV reduction (transformation from the ECDLP to the DLP in a finite field) is completed in probabilistic polynomial time in $k \log q$ ($k \le 6$),

and so the MOV algorithm for supersingular curves is completed in subexponential time in $\log q$. However, there exist two major problems to clear, from an algorithmic point of view, in applying the MOV reduction to general elliptic curves (assuming that $n = \#\langle P \rangle$ is prime number and n/q):

- 1. the problem of explicitly determining the smallest positive integer k such that $E[n] \subset E(F_{g^k})$, where E[n] is the set of n-torsion points.
- 2. the problem of efficiently finding an n-torsion point Q such that $e_n(P,Q)$ has order n (i.e. $e_n(P,Q) \neq 1$ because of the assumption that n is a prime number.), where e_n is the Weil pairing. (In the sequel, we refer to such an n-torsion point Q as a "good" n-torsion point.)

For the first problem, we can find an answer to it in a paper by Balasubramanian and Koblitz [3]. They proved that if n/q-1, k is the smallest positive integer such that $q^k \equiv 1 \mod n$. (It is interesting to note that this condition is identical to the one under which the FR reduction is applied.) In the same paper, they also suggest that we need k=n if n|q-1 and $E[n]/E(F_q)$. Thus, when n is much larger than $\log q$, we may give up applying the MOV algorithm since the extension degree in this case is too large in order for the reduced DLP in $F_{q^k}^*$ to be solved in subexponential time in $\log q$.

For the second problem, we cannot find any answer which covers all the case: a simple and widely well-known method generally requires exponential time in $\log q$ even if k is small (Section 4.1). Moreover, the methods using the multiplication by constant maps in a suitable way might also take exponential time in $\log q$ for the general case (Section 4.2).

Thus, in order to reach the valid conclusion that the MOV algorithm is always completed in subexponential time in $\log q$ if the DLP in $F_{q^k}^*$ is solved in subexponential time in $\log q$, an efficient method which solve the second problem above will be desired.

1.2 Main Result

The major contribution of this paper is to solve the second problem described earlier by constructing a novel method which finds a "good" n-torsion point required in evaluating the Weil pairing in probabilistic polynomial time in $k \log q$, under the most reasonable assumptions stated above (i.e. n is a prime such that n/q| and n/q|-1). This expected running time is optimal, since it always means probabilistic polynomial time in $\log q$ whenever k is small enough to solve the DLP in F_{qk}^* in subexponential time in $\log q$. As a result, we obtain an optimized MOV algorithm for general elliptic curves.

The key idea which leads us to successfully finding a "good" n-torsion point efficiently is to construct a homomorphism $f: E(F_{q^k}) \to E(F_{q^k})$ such that Im f = E[n]. We will see that it is possible by using the q-th power Frobenius map under a certain condition.

Now, we turn our attention to comparing the MOV and FR reductions. It may have been believed by some cryptographers that assuming n/q|-1, the MOV reduction is as powerful as the FR reduction in the sense that their minimum extension degrees k coincide when the base field F_q is extended to F_{q^k} in order to apply those reductions. However, so far there has been a lack of a formal proof that supports the belief. As pointed out in [12], the problem of efficiently finding a "good" n-torsion point required in evaluating the Weil pairing should be solved as well. Thus, our contribution allows us to finally draw the conclusion that the MOV reduction is indeed as powerful as the FR reduction under n/q|-1, in a true sense: not only from the viewpoint of the minimum extension degree of the base field but also from that of the effectiveness of algorithms.

The rest of this paper is organized as follows: In Section 2, we briefly review some basic facts on elliptic curves over finite fields and the MOV algorithm. In Section 3, we consider the problem of explicitly determining the minimum extension degrees and describe the answer to it obtained by Balasubramanian and Koblitz. In Section 4, we consider the problem of efficiently finding a "good" n-torsion point. Three different methods are considered to solve the problem. The third method is completed in probabilistic polynomial time in $k \log q$ for the general case n/q-1. Finally, based on the efficient method in the previous section, in Section 5 we actually realize an optimized MOV algorithm for general elliptic curves under n/q-1 and estimate its running time.

2 Preliminaries

In this section, we briefly review some materials on elliptic curves over finite fields. (See [27] for more details.)

Let F_q be a finite field with $q=p^m$ elements, where p is a prime number, and \bar{F}_q its algebraic closure. Let E be an elliptic curve over F_q given by the Weierstrass equation

$$E: y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6$$
 (1)

whose coefficients lie in F_q . For each extension field K of F_q , E(K) is given by

$$E(K) = \{(x, y) \in K \times K | (x, y) \text{ satisfies } (1) \} \cup \{O\}$$

where O is a special point, called the point at infinity. There is an abelian group structure on the points of E(K), in which O serves as its identity element, given by the so-called tangent-and-chord method. We express its abelian structure additively.

Let n be a positive integer relatively prime to p, the characteristic of F_q . The Weil pairing is a map

$$e_n: E[n] \times E[n] \longrightarrow \mu_n \subset \bar{F}_q$$

where $E[n] = \{T \in E(\bar{F}_q) | [n]T = O\}$ is the group of *n*-torsion points and μ_n is the subgroup of *n*-th roots of unity in \bar{F}_q . For properties of the Weil pairing, see [27] [18].

Let $P \in E[n]$ be a point of order n. Then, we have the following:

Proposition 1. ([27][19]) There exists $Q \in E[n]$ such that $e_n(P,Q)$ is a primitive n-th root of unity. Therefore,

$$f_Q: \langle P \rangle \longrightarrow \mu_n, \quad f_Q(S) = e_n(S, Q)$$

is a group isomorphism.

Based on this fact, the framework of the MOV algorithm can be described as follows:

Algorithm 1 ([18] [19])

Input: An element $P \in E(F_q)$ of order n, and $R \in \langle P \rangle$.

Output: An integer l such that R = [l]P.

Step 1: Determine the minimum positive integer k such that $E[n] \subset E(F_{a^k})$.

Step 2: Find $Q \in E[n]$ such that $\alpha = e_n(P,Q)$ has order n.

Step 3: Compute $\beta = e_n(R, Q)$.

Step 4: Compute l, the discrete logarithm of β to the base α in $F_{q^k}^*$.

This algorithm is somewhat incomplete in that the methods for determining k and for finding a point Q are not explicitly given. For supersingular elliptic curves, the methods which settle those problems are given in [19]; the resulting minimum k are k=1,2,3,4, or 6, and for each corresponding k,Q is efficiently obtained by using the group structure of $E(F_{q^k})$. Therefore, for supersingular elliptic curves, the reduction is completed in probabilistic polynomial time in $\log q$ and the algorithm mentioned above takes probabilistic subexponential time in $\log q$.

In the following sections, we consider the two problems described in Section 1 not only for the supersingular case but also for the non-supersingular (ordinary) case .

3 Determining the Minimum Extension Degrees

In this section, we consider the problem of determining the minimum positive integer k such that $E[n] \subset E(F_{a^k})$.

The following proposition is proved by Schoof [24].

Proposition 2. ([24]) Let p be the characteristic of F_{q^k} , n a natural number with p / n and t_k denote the trace of the q^k -th power Frobenius map ϕ of E. The following are equivalent;

 $(1) \ E[n] \subset E(F_{q^k})$

(2)
$$n^2 | \#E(F_{q^k}), \ n | q^k - 1 \ and \ either \ \phi \in Z \ or \ \mathcal{O}(\frac{t_k^2 - 4q^k}{n^2}) \subset End_{F_{q^k}}(E)$$

where $\mathcal{O}(\frac{t_k^2-4q^k}{n^2})$ is an order of discriminant $\frac{t_k^2-4q^k}{n^2}$.

However, from an algorithmic point of view, a more explicit form of k is needed to realize the MOV reduction. With the assumption that n is a prime number such that $n|\#E(F_q)$ and n/q|-1, Balasubramanian and Koblitz [3] have obtained the following result:

Proposition 3. ([3]) Let E be an elliptic curve defined over F_q , and suppose that n is a prime number such that $n|\#E(F_q)$, n/q|-1. Then, $E[n] \subset E(F_{q^k})$ if and only if $n|q^k-1$.

Remark 1. It is important to note that Balasubramanian and Koblitz's results also suggest k=n if n|q-1 and E[n] / $\mathcal{E}(F_q)$. Thus, in this case, when n is much larger than $\log q$, we may give up applying the MOV algorithm since the extension degree in this case is too large in order for the reduced DLP in $F_{q^k}^*$ to be solved in subexponential time in $\log q$.

4 Three Methods for Finding *n*-Torsion Points

In this section, we consider the problem of finding an n-torsion point $Q \in E[n]$ such that $\alpha = e_n(P, Q)$ has order n. (See Algorithm 1 in Section 2.) We refer to such an n-torsion point Q as a "good" n-torsion point.

As before, we assume the following:

Assumption 1 (1) n is a prime number; (2) n /q; (3) n /q| - 1.

The first condition is not restrictive, since we can reduce the composite case to the prime one by applying the Chinese Remainder Theorem and the Pohlig-Hellman algorithm; the second one is necessary, since the Weil pairing is not defined otherwise; the third one is reasonable from the result by Balasubramanian and Koblitz. (See Remark in Section 3.)

Also, as before, we use the following notation: P is a base point of order n. (Thus, $E(F_q)[n] = \langle P \rangle \cong Z/nZ$.); k is the minimum positive integer such that $E[n] \subset E(F_{q^k})$, or equivalently k is the minimum positive integer with $n|q^k-1$. (See Proposition 3 in Section 3.)

Let N_k be the number of F_{q^k} -rational points on E, and $E(F_{q^k})_n$ the n-primary part of $E(F_{q^k})$, i.e.

$$N_k = \#E(F_{q^k}), \qquad E(F_{q^k})_n = \bigcup_{i > 1} E(F_{q^k})[n^i],$$

and, let $d = v_n(N_k)$ denote the largest integer such that $n^d|N_k$.

Now, we provide three different methods to find a "good" n-torsion point; the first one, which is considerable simple, repeatedly chooses $Q \in E(F_{q^k})$ until both $Q \in E[n]$ and $e_n(P,Q) \neq 1$ are satisfied; the second one is a method using the multiplication by constant maps and can be regarded as a generalized version of the algorithm that Menezes, Okamoto, and Vanstone considered in the original paper [19] on the MOV reduction for supersingular elliptic curves; and the third one, which is constructed based on Theorem 1 given later, can be applied to

the general case n/q-1 and is completed in probabilistic polynomial time in $k \log q$. It turns out that the second one takes a smaller expected number of iterations than the first one in order to obtain a "good" n-torsion point. However, they generally require exponential time in $\log q$. The third one, our final goal of this section, is optimal, since it is completed in probabilistic polynomial time in $k \log q$ for general elliptic curves.

4.1 A Simple Method

We assume that $E(F_{q^k}) \cong Z/cd_1nZ \times Z/cnZ$ ($cn|q^k-1$). (Note that the group structure of $E(F_{q^k})$ can be always expressed in this form [18].)

The first method is simple:

Procedure 1

Step 1: Choose $Q \in E(F_{q^k})$ randomly.

Step 2: Check if $Q \in E[n]$ by computing [n]Q. If $Q \not\in E[n]$, go to Step 1.

Step 3: Compute $\alpha = e_n(P,Q)$. If $\alpha = 1$, go to Step 1.

In Step 1, we first pick an element x = a in F_{q^k} to substitute it to the equation (1). Then we check if the quadratic equation with respect to y has a solution in F_{q^k} . If it does, we solve the quadratic equation in a usual manner. (See, for example, [15][18] for the details.) Also, for Step 3, there is a standard procedure to compute the Weil pairing. (See, for example, [18].) Note that we can execute this method even if the group structure of $E(F_{q^k})$ is unknown.

If Procedure 1 is applied, the probability of finding a "good" point Q for each iteration is

$$\frac{\#E[n]}{\#E(F_{q^k})} \times \frac{\#E[n] - \#\langle P \rangle}{\#E[n]} = \frac{n^2}{d_1c^2n^2} \times \frac{n^2 - n}{n^2} = \frac{1}{d_1c^2}(1 - \frac{1}{n}).$$

Thus, the success probability for each iteration is approximately $\frac{1}{d_1c^2}$ since n is assumed to be large enough. If n = O(q), the expected number of iterations is approximately $d_1c^2 = N_k/n^2 = O(q^{k-2})$, where for the last equality n = O(q) and the Hasse bound [27] have been applied. Therefore, if k > 2, the above method is no longer efficient, since it takes exponential time in $\log q$.

4.2 Methods Using the Multiplication by Constant Maps

The second method is a generalized version of that considered in the original paper [19] on the MOV reduction for supersingular elliptic curves. Two versions of this method are considered. One may use one of these versions depending on whether the knowledge of the group structure of $E(F_{q^k})$ is required or not (Procedure 2 and 3, respectively). Procedure 2 is considered in [12]. However, Procedure 3 is different from the method in [12], since our method does not need the information of the complete group structure of $E(F_{q^k})$.

We first consider the case that the group structure $E(F_{q^k}) \cong Z/cd_1nZ \times Z/cnZ$ $(cn|q^k-1)$ is known.

Procedure 2

Step 1: Set $v_n(d_1)$ the largest integer such that $n^{v_n(d_1)}|d_1$ and set $d_2 := d_1/n^{v_n(d_1)}$.

Step 2: Choose $Q \in E(F_{q^k})$ randomly.

Step 3: Set $Q' = [cd_2]Q \in E[n^{v_n(d_1)+1}] \cap E(F_{q^k}) \cong Z/n^{v_n(d_1)+1} \times Z/n$.

Step 4: Check if $Q' \in E[n]$ by computing [n]Q'. If $Q' \not \in E[n]$, go to Step 2.

Step 5: Compute $\alpha = e_n(P, Q')$. If $\alpha = 1$, go to Step 2.

If Procedure 2 is applied, the probability of finding a "good" point Q' for each iteration is

$$\frac{\#E[n]}{\#(E[n^{v_n(d_1)+1}]\cap E(F_{q^k})}\times \frac{\#E[n]-\#\langle P\rangle}{\#E[n]} = \frac{n^2}{n^{v_n(d_1)+2}}\times \frac{n^2-n}{n^2} = \frac{1}{n^{v_n(d_1)}}(1-\frac{1}{n})$$

In particular, if n / d_1 , this method is simplified to:

Procedure 2'

Step 1: Choose $Q \in E(F_{a^k})$.

Step 2: Set $Q' = [cd_1]Q \in E[n]$.

Step 3: Compute $\alpha = e_n(P, Q')$. If $\alpha = 1$, go to Step 1.

The probability of finding a "good" point Q' for each iteration is 1 - 1/n. Note that, for supersingular elliptic curves, $d_1 = 1$ and this method coincides with what was used in [19]. In this sense, Procedure 2 can be regarded as a generalized version of that used in [19].

The probability of finding a "good" point for each iteration in Procedure 2 is approximately $1/n^{v_n(d_1)}$, and the expected number of iterations is approximately $n^{v_n(d_1)}$, which is smaller than that of Procedure 1 since $n^{v_n(d_1)} \leq d_1 \leq d_1 c^2$.

We next consider the case that the group structure of $E(F_{q^k})$ is unknown beforehand. For finding the group structure of $E(F_{q^k})$, we apply Miller's algorithm, which finds the pair (n_1, n_2) such that $E(F_{q^k}) \cong Z/n_1Z \times Z/n_2Z$ $(n_2|n_1, n_2|q^k-1)$ assuming the knowledge of the factorization of N_k . (For the details of Miller's algorithm, see [18] [19].) However, since we are looking at the n-primary part, all we need is the information on the group structure of that, i.e. the pair (r,s) such that $E(F_{q^k})_n \cong Z/n^rZ \times Z/n^sZ$ $(1 \le s \le r)$. Then, we can avoid computing the factorization of N_k , and consequently that leads to a great saving of computation. Thus, in the following procedure (Procedure 3), we make use of a simplified version of Miller's algorithm $(N_*Miller)$ which computes the group structure of the n-primary part without the knowledge of the factorization of N_k . The essential difference be!! tween Procedures 2 and 3 lies in this point.

Procedure 3

Step 1: Compute $N_1 = \#E(F_q)$.

Step 2: Compute $N_k = \#E(F_{q^k})$ from $N_1 = \#E(F_q)$, using the Weil Theorem, and $d = v_n(N_k)$.

Step 3: Execute N_Miller to get the pair (r, s).

Step 4: Compute $t = N_k/n^{r+1}$.

Step 5: Choose $Q \in E(F_{q^k})$ randomly, and compute Q' = [t]Q.

Step 6: Check if $Q' \in E[n]$ by computing [n]Q'. If $Q' \not\in E[n]$, go to Step 5.

Step 7: Compute $\alpha = e_n(P, Q')$. If $\alpha = 1$, go to Step 5.

 $N_Miller:$

1) $Pick\ V, W \in E(F_{q^k})\ randomly.$

2) Compute $V' = [N_k/n^d]V$ and $W' = [N_k/n^d]W$.

3) Compute ord(V'), ord(W') (the orders of V', W', respectively), and set $r = max\{v_n(ord(V')), v_n(ord(W'))\}$.

4) Compute $\delta = e_{n^r}(V', W')$ and its order $n^s = ord(\delta)$.

5) If r + s = d, then return (r, s). Otherwise, go to 1).

We provide explanation of each step in the above method.

In Step 1, we compute N_1 in polynomial time, using the Schoof-Elkies-Atkin algorithm and its variants [25][5][16][6][1][2][9] [22][17][7][13].

As described earlier, N-Miller is regarded as Miller's algorithm that finds the group structure of the n-primary part $E(F_{q^k})_n$ of $E(F_{q^k})$. In 2), note that the multiplication by N_k/n^d map $[N_k/n^d]: E(F_{q^k}) \to E(F_{q^k})$ is an abelian group homomorphism and hence it preserves the uniform distribution. Moreover, since its image is the n-primary part $E(F_{q^k})_n$, we can obtain $V', \ W' \in E(F_{q^k})_n$ randomly if we pick $V, \ W \in E(F_{q^k})$ randomly. In 5), if r+s=d, we can see that the group structure of the n-primary part is isomorphic to $Z/n^rZ \times Z/n^sZ$, and also the probability of success is

$$\frac{\varphi(n^r)\varphi(n^s)}{n^{r+s}} = \frac{n^{r-1}(n-1)n^{s-1}(n-1)}{n^{r+s}} = (1 - \frac{1}{n})^2,$$

where φ is the Euler function.

In Steps 4 and 5, we note that $t = \frac{N_k}{n^{r+1}} = \frac{N_k}{n^d} \cdot n^{s-1}$. Therefore, the image of the multiplication by t map $[t] = [n^{s-1}] \circ [N_k/n^d] : E(F_{q^k}) \longrightarrow E(F_{q^k})$ is exactly isomorphic to $Z/n^{r-s+1}Z \times Z/nZ$. Thus [t] is an analogue of $[cd_2]$ in Step 3 of Procedure 2, although [t] is not correctly corresponding to $[cd_2]$.

Finally, we briefly analyze the time complexity of our method. (i.e. Procedure 2 and 3.) From the considerations described earlier, it follows that the success probability for each iteration is approximately $1/n^{v_n(d_1)}$ and the expected number of iterations is $O(n^{v_n(d_1)})$. Thus, if $v_n(d_1)=0$, i.e. $d=v_n(N_k)$ is even and the group structure of the n-primary part $E(F_{q^k})_n$ is isomorphic to $Z/n^{\frac{d}{2}}Z \times Z/n^{\frac{d}{2}}Z$, they are completed in probabilistic polynomial time in $k \log q$. Otherwise, if n is exponential in $\log q$, they are no longer efficient.

Remark 2. It is possible to improve the method in order to make the success probability high: after picking a point $Q \in E(F_{q^k})_n$ randomly by using the map $[N_k/n^d]$, we compute its order, say, n^l . Then we can obtain $Q' = [n^{l-1}]Q \in E[n]$. This might be familiar to some people. The success probability of this method is better than that of the above methods. However, in general cases, this procedure also requires the expected number of iterations $O(n^{v_n(d_1)})$ and the time complexity remains same. More precisely, we cannot reduce the expected

running time when $\langle P \rangle = \langle n^{r-1}S \rangle$, where $E(F_{q^k})_n \cong Z/n^rZ \times Z/n^sZ$ ($1 \le s < r$) and S is an element of order n^r . Otherwise, the expected number of iterations is almost one. In fact, this is clear by considering the two cases: $E(F_{q^k})_n \cong Z/n^sZ \times Z/n^sZ$; $E(F_{q^k})_n \cong Z/n^rZ \times Z/n^sZ$ ($1 \le s < r$) with $\langle P \rangle \neq \langle n!!^{r-1}S \rangle$. The first case has been already considered in the previous methods while the second case will be addressed in the following section. The above case (i.e. $\langle P \rangle = \langle n^{r-1}S \rangle$) is essentially solved by proposing the next method in Section 4.3.

Example 1. As an example that the methods in this subsection are not efficient, we can consider the case that $E(F_q)_n \cong Z/n^2Z$, $E(F_{q^k})_n \cong Z/n^2Z \times Z/nZ$, where n is exponential in $\log q$.

4.3 An Efficient Method for General Elliptic Curves.

The methods described before are not always completed in polynomial time in $k \log q$ for the general case. In other words, we need some assumptions in order for them to be completed in polynomial time in $k \log q$. Thus, we consider to remove this restriction in this subsection. The key idea is to construct a homomorphism $f: E(F_{q^k}) \to E(F_{q^k})$ such that $\mathrm{Im} f = E[n]$, and we will see that it is possible by using the q-th power Frobenius map ϕ under a certain condition.

As a natural situation, we assume that the group structure of $E(F_{q^k})$ is unknown beforehand. The following is our proposed method for general elliptic curves.

Procedure 4

```
Step 1: Compute N_1 = \#E(F_q).
```

Step 2: Compute $N_k = \#E(F_{q^k})$ from N_1 , and $d = v_n(N_k)$.

Step 3: Execute N_Miller to obtain the pair (r, s). If r = s, go to Step 5. **Step 4:**

(4-1) Choose $Q \in E(F_{a^k})$ randomly.

(4-2) Compute $Q' = [N_k/n^{s+1}]Q$. If Q' = O, go to (4-1). Otherwise, compute $Q'' = (\phi - 1)Q'$.

(4-3) If $Q'' \neq O$, compute $\alpha = e_n(P, Q')$ and go to Step 6.

Step 5:

(5-1) Choose $Q \in E(F_{q^k})$ randomly.

(5-2) Compute $Q' = (\phi - 1)^{r-s} \circ [N_k/n^{r+1}]Q$. (We define $(\phi - 1)^0 := id$: identity map.)

(5-3) Compute $\alpha = e_n(P, Q')$. If $\alpha = 1$, go to (5-1).

Step 6: Store Q' and α .

We provide explanation of each step in the above method.

In Step 3, we can know the group structure of the n-primary part $E(F_{q^k})_n \cong Z/n^r Z \times Z/n^s Z$ ($1 \leq s \leq r$). If r = s, the rest of this method is same as Procedure 3, which is proposed in Section 4.2.

In Step 4, we assume that $E(F_{q^k})_n = \langle S \rangle \times \langle T \rangle \cong Z/n^r Z \times Z/n^s Z$ (1 $\leq s < r$), where S and T are generators of orders n^r and n^s , respectively. The

image of the multiplication by $N_k/n^{s+1} = n^{r-1} \times N_k/n^d$ map $[n^{r-1}] \circ [N_k/n^d]$: $E(F_{q^k}) \to E(F_{q^k})$ is isomorphic to Z/nZ. We can know whether $\langle P \rangle = \langle n^{r-1}S \rangle$ or not by checking whether Q'' = O or not. In fact, since the order of Q' is n and $Q'(\neq O) \in \langle n^{r-1}S \rangle$, and since we have assumed $E(F_q)[n] = \langle P \rangle$, it follows that $\langle P \rangle = \langle n^{r-1}S \rangle \Leftrightarrow Q' \in \langle P \rangle \Leftrightarrow (\phi-1)Q' = O$. In order to check whether $\langle P \rangle = \langle n^{r-1}S \rangle$, we need $Q' \neq O$, and its success probability is

$$\frac{\varphi(n^r)n^s}{n^{r+s}} = \frac{n^{r-1}(n-1)n^s}{n^{r+s}} = 1 - \frac{1}{n}$$

if we choose $Q \in E(F_{q^k})$ randomly. When $\langle P \rangle \neq \langle n^{r-1}S \rangle$, we can obtain $\alpha = e_n(P,Q') \neq 1$.

In Step 5, we already know that $E(F_{q^k})_n \cong Z/n^sZ \times Z/n^sZ$, or $E(F_{q^k})_n = \langle S \rangle \times \langle T \rangle \cong Z/n^rZ \times Z/n^sZ$ ($1 \leq s < r$) with $\langle P \rangle = \langle n^{r-1}S \rangle$. For the first case, the rest of the method is same as Procedure 3. Therefore, when r = s, the success probability for each iteration is 1 - 1/n. For the second case, in order to explain the validity of this step, we need the following theorem:

Theorem 1. We assume that

$$E(F_{q^k})_n = \langle S \rangle \times \langle T \rangle \cong Z/n^r Z \times Z/n^s Z \quad (1 \le s < r),$$
$$\langle P \rangle = \langle n^{r-1} S \rangle,$$

where S and T are generators of orders n^r and n^s , respectively. Consider the map

$$f=(\phi-1)^{r-s}\circ [n^{s-1}]\circ [N_k/n^d]:\ E(F_{q^k})\longrightarrow E(F_{q^k}).$$

Then we have

$$\mathrm{Im} f \cong Z/nZ \times Z/nZ.$$

Proof. Consider the multiplication by N_k/n^d map $[N_k/n^d]: E(F_{q^k}) \longrightarrow E(F_{q^k})$. It is easy to see that its image is the n-primary part $E(F_{q^k})_n$ of $E(F_{q^k})$. We define $f^{(r-s)}:=(\phi-1)^{r-s}: E(F_{q^k}) \longrightarrow E(F_{q^k})$, where ϕ is the q-th power Frobenius map. Then, from Lemma 1, which will be given below, it follows that $\operatorname{Im}(f^{(r-s)}\circ [N_k/n^d])=\operatorname{Im}(f^{(r-s)}|_{E(F_{q^k})_n})\cong Z/n^sZ\times Z/n^sZ$. Moreover, by composing the map $[n^{s-1}]$ with it, we have $\operatorname{Im}([n^{s-1}]\circ f^{(r-s)}\circ [N_k/n^d])\cong Z/nZ\times Z/nZ$. Therefore, $\operatorname{Im} f\cong Z/nZ\times Z/nZ$ follows, since

$$[n^{s-1}] \circ f^{(r-s)} \circ [N_k/n^d] = f^{(r-s)} \circ [n^{s-1}] \circ [N_k/n^d]. \qquad \square$$

Lemma 1. We assume that

$$E(F_{a^k})_n = \langle S \rangle \times \langle T \rangle \cong Z/n^r Z \times Z/n^s Z \quad (1 \le s < r),$$

where S and T are generators of orders n^r and n^s , respectively, and

$$\langle P \rangle = \langle n^{r-1} S \rangle.$$

We consider the map:

$$f^{(i)} = (\phi - 1)^i : E(F_{q^k}) \longrightarrow E(F_{q^k}) \ (0 \le i \le r - s),$$

where ϕ is the q-th power Frobenius map. Then we have

$$\operatorname{Im}(f^{(i)}|_{E(F_{a^k})_n}) \cong Z/n^{r-i}Z \times Z/n^sZ$$

and $\text{Im}(f^{(i)}|_{E(F_{q^k})_n})$ is generated by $f^{(i)}(S)$ and $f^{(i)}(T)$ of orders n^{r-i} and n^s , respectively.

Proof. The proof is given in Appendix A. \Box

From Theorem 1, it follows that $Q' \in E[n]$ in Step~(5-2), since $(\phi-1)^{r-s} \circ [N_k/n^{r+1}] = (\phi-1)^{r-s} \circ [n^{s-1}] \circ [N_k/n^d]$. Moreover, when r > s, the effectiveness of Step~5 is justified by the following proposition:

Proposition 4. We assume that

$$E(F_{q^k})_n = \langle S \rangle \times \langle T \rangle \cong Z/n^r Z \times Z/n^s Z \quad (1 \le s < r)$$
$$\langle P \rangle = \langle n^{r-1} S \rangle,$$

where S and T are generators of orders n^r and n^s , respectively. Then, in Step 5 of Procedure 4, the probability of obtaining $\alpha /=$ is 1-1/n.

Proof. Since $\frac{N_k}{n^{r+1}} = n^{s-1} \cdot \frac{N_K}{n^d}$, we have

$$f = (\phi - 1)^{r-s} \circ [N_k/n^{r+1}] = (\phi - 1)^{r-s} \circ [n^{s-1}] \circ [N_k/n^d].$$

The map $f: E(F_{q^k}) \longrightarrow E(F_{q^k})$ is an abelian group homomorphism and it preserves the uniform distribution. Moreover, its image is isomorphic to $Z/nZ \times Z/nZ$. (See Theorem 1.) Thus the probability of finding $Q \in E(F_{q^k})$ such that $e_n(P, f(Q)) /= 1$ is

$$\frac{\#E[n] - \#\langle P \rangle}{\#E[n]} = \frac{n^2 - n}{n^2} = 1 - \frac{1}{n}.$$

Finally, from the considerations above, it follows that the probability of success in Procedure 4 is approximately one and that it is completed in probabilistic polynomial time in $k \log q$. (See Section 5.2. for more details.)

5 Optimizing the MOV Algorithm for General Elliptic Curves

In this section, we actually realize the MOV algorithm for general elliptic curves under Assumption 1, using the results obtained in the previous sections.

5.1 Description of an Optimized MOV Algorithm

The MOV algorithm is completed, based on the results in previous sections, as follows:

Algorithm 2 (An Optimized MOV Algorithm)

```
Input: An elliptic curve E, a base point P \in E(F_q) and R \in \langle P \rangle.
Output: An integer l such that R = [l]P.
Step 1: Determine the minimum positive integer k such that q^k \equiv 1 \mod n.
Step 2:
    (2-1) Compute N_1 = \#E(F_q).
    (2-2) Compute N_k = \#E(F_{q^k}) from N_1, and d = v_n(N_k).
    (2-3) Execute N_Miller to obtain the pair (r, s). If r = s, go to (2-5).
    (2-4):
        (2-4-1) Choose Q \in E(F_{q^k}) randomly.
        (2-4-2) Compute Q' = [\hat{N}_k/n^{s+1}]Q. If Q' = O, go to (2-4-1). Other-
            wise, compute Q'' = (\phi - 1)Q'.
        (2-4-3) If Q'' \neq O, compute \alpha = e_n(P, Q') and go to (2-6).
    (2-5):
        (2-5-1) Choose Q \in E(F_{q^k}) randomly.
        (2-5-2) Compute Q' = (\dot{\phi} - 1)^{r-s} \circ [N_k/n^{r+1}]Q.
        (2-5-3) Compute \alpha = e_n(P, Q'). If \alpha = 1, go to (2-5-1).
    (2-6) Store Q' and \alpha.
```

Step 3: Compute $\beta = e_n(R, Q')$.

Step 4: Compute l, the discrete logarithm of β to the base α in $F_{a^k}^*$.

 $N_Miller:$

- 1) $Pick\ V, W \in E(F_{a^k})\ randomly.$
- **2)** Compute $V' = [N_k/n^d]V$ and $W' = [N_k/n^d]W$.
- 3) Compute ord(V'), ord(W') and set $r = max\{v_n(ord(V')), v_n(ord(W'))\}$.
- 4) Compute $\delta = e_{n^r}(V', W')$ and its order $n^s = ord(\delta)$.
- 5) If r + s = d, then return (r, s). Otherwise, go to 1).

5.2 Success Probability and Running Time

We consider the success probability and running time of Algorithm 2.

- Success Probability:
 - 1. the success probability in *N_Miller* is approximately $(1 1/n)^2$. (See Section 4.2.)
 - 2. the success probability in Step~(2-4) is approximately 1-1/n. (See Section 4.3.)
 - 3. the success probability in Step~(2-5) is approximately 1-1/n. (See Section 4.3.)

Therefore, once we determine the value k in $Step\ 1$, the success probability of the rest of the reduction (i.e. $Step\ 2$ and $Step\ 3$) is approximately one.

- Running Time:

We assume that the usual multiplication algorithms are used, so that multiplying two elements of length N takes time $O(N^2)$. We estimate the running time of the following major computation.

- 1. Computation of $\#E(F_q)$ using the Schoof-Elkies-Atkin algorithm and its variants (in $Step\ (2-1)$): this procedure requires $O(\log^6 q)$.
- 2. Picking a random point on $E(F_{q^k})$: this procedure requires $O(k^3 \log^3 q)$.
- 3. Computation of Q' (and V', W'): computation of Q' in Step~(2-4-2) requires $O((\log N_k)(k\log q)^2) = O((k\log q)(k\log q)^2) = O(k^3\log^3 q)$, where $N_k = \#E(F_{q^k})$. Similarly, computation of V' and W' requires $O(k^3\log^3 q)$. Computation of Q' in Step~(2-5-2) requires $O(k^3\log^3 q + (r-s)(\log q)(k\log q)^2) = O(k^3\log^3 q)$.
- 4. Computation of the Weil pairing $e_n(P,Q')$: this procedure requires $O(k^3 \log^3 q + (\log n)(k \log q)^2) = O(k^3 \log^3 q + k^2 \log^3 q) = O(k^3 \log^3 q)$. Similarly, computation of the Weil pairing $e_{n^r}(V',W')$ in N_Miller requires $O(k^3 \log^3 q + (r \log n)(k \log q)^2) = O(k^3 \log^3 q)$.

Also, each procedure in $Step\ 2$ and $Step\ 3$, except the above, require at most $O(k^3\log^3q)$. Therefore, once we determine the value k in $Step\ 1$, the rest of the reduction (i.e. $Step\ 2$ and $Step\ 3$) is completed in polynomial time in $k\log q$, more precisely, $O(k^3\log^3q + \log^6q)$.

Acknowledgements

The authors would like to thank Nigel Smart for useful comments on the earlier version of this paper, Shigenori Uchiyama and Taiichi Saito for fruitful discussion at some domestic conferences and the referees for useful comments.

References

- A. O. L. Atkin, "The number of points on an elliptic curve modulo a prime", Draft, 1988.
- A. O. L. Atkin, "The number of points on an elliptic curve modulo a prime (ii)," Draft, 1992. 94
- R. Balasubramanian and N. Koblitz, "The improbability that an elliptic curve has subexponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm," in Journal of Cryptology 11, pp.141-145, 1998. 88, 91
- H. Cohen, "A Course in Computational Algebraic Number Theory," Springer-Verlag, Berlin, 1993.
- J.-M. Couveignes and F. Morain, "Schoof's algorithm and isogeny cycles," in the Proc. of ANTS-I, Lecture Notes in Computer Science 877, pp.43-58, 1994.
- J.-M. Couveignes, L. Dewaghe and F. Morain, "Isogeny cycles and the Schoof-Elkies-Atkin algorithm," LIX/RR/96/03, 1996.
 94
- J.-M. Couveignes, "Computing l-isogenies using the p-torsion," in the Proc. of ANTS-II, Leture Notes in Computer Science 1122, pp.59-65, Springer, 1996.
- 8. T. Denny, O. Schirokauer and D. Weber, "Discrete logarithms: the effectiveness of the index calculus method," in the Proc. of ANTS-II, Lecture Notes in Computer Science 1122, pp.337-362, Springer-Verlag, 1996. 87

- 9. N. D. Elkies, "Explicit isogenies," Draft, 1991. 94
- G. Frey and H. G. Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves," Math. Comp., 62, 206, pp.865-874, 1994.
- G. Frey, M. Müller and H. G. Rück, "The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems," preprint, 1998.
- R. Harasawa, J. Shikata, J. Suzuki and H. Imai, "Comparing the MOV and FR reductions in elliptic curve cryptography," Advances in Cryptology - EURO-CRYPT'99, Lecture Notes in Computer Science, vol.1592, pp.190-205, 1999. 87, 89, 92
- T. Izu, J. Kogure, M. Noro, and K. Yokoyama, "Efficient Implementation of Schoof's Algorithm," Advances in Cryptology - ASIACRYPT'98, Lecture Notes in Computer Science 1514, Springer, pp.66-79, 1998.
- N. Koblitz, "Elliptic Curve Cryptosystems," Math. Comp. 48, pp.203-209, 1987.
- 15. N. Koblitz, "Algebraic Aspects of Cryptography," Springer-Verlag, 1998. 92
- R. Lercier and F. Morain, "Counting the number of points on elliptic curves over finite fields: strategy and performances," Advances in Cryptology - EURO-CRYPT'95, Lecture Notes in Computer Science, vol.921, pp.79-94, 1995.
- 17. R. Lercier, "Computing isogenies in F_{2^n} ," In the Proc. of ANTS-II, Lecture Notes in Computer Science 1122, Springer, pp.197-212, 1996. 94
- A. Menezes, "Elliptic Curve Public Key Cryptosystem," Kluwer Acad. Publ., Boston, 1993. 89, 90, 92, 93
- A. Menezes, T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms in a finite field," IEEE Transactions on Information Theory, vol.IT-39, no.5, pp.1639-1646, 1993. 87, 90, 91, 92, 93
- V. Miller, "Use of elliptic curves in cryptography," Advances in Cryptology -CRYPTO'85, Lecture Notes in Computer Science, vol.218, pp.417-426, Springer, 1986.
- V. Miller, "Short programs for functions on curves," unpublished manuscript, 1986.
- F. Morain, "Calcul du nombre de points sur une courbe elliptique dans un corps fini : aspects algorithmiques," J. Théor. Nombres Bordeaux 7, pp.255- 282, 1995..
- T. Satoh and K. Araki, "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves," Commentarii Math. Univ. St. Pauli, 47(1), pp.81-92, 1998.
- 24. R. Schoof, "Nonsingular plane cubic curves over finite fields," J. Combinatorial Theory, Series A, vol.46, pp.183-211, 1987. 90
- R. Schoof, "Counting points on elliptic curves over finite fields," J. Théor. Nombres Bordeaux 7, pp.219-254, 1995.
- 26. I. Semaev, "Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p," Math. of Computation 67, pp.353-356, 1998. 87
- J. Silverman, "The Arithmetic of Elliptic Curves," Springer-Verlag, New York, 1986. 89, 90, 92
- 28. J. Silverman and J. Suzuki, "Elliptic curve discrete logarithms and index calculus," Advances in Cryptology ASIACRYPT'98, Lecture Notes in Computer Science 1514, Springer, pp.110-125, 1998.
- N. Smart, "The Discrete logarithm problem on elliptic curves of trace one," to appear in Journal of Cryptology.

Appendix A (Proof of Lemma 1)

Proof of Lemma 1. The proof is completed by induction on i.

The case i=0 is trivial. We consider the case i=1. Set $S':=f^{(1)}(S)=(\phi-1)S$ and $T':=f^{(1)}(T)=(\phi-1)T$. We first show that the orders of S' and T' are n^{r-1} and n^s , respectively. Clearly, $n^{r-1}S'=(\phi-1)(n^{r-1}S)=O$. Since

$$n^j S' = O \Leftrightarrow (\phi - 1)(n^j S) = O \Leftrightarrow n^j S \in \langle P \rangle = \langle n^{r-1} S \rangle,$$

it follows that $j \geq r-1$. Thus, the order of S' is n^{r-1} . Also, clearly, $n^sT' = (\phi - 1)(n^sT) = O$. To prove that the order of T' is n^s , it is sufficient to show that $n^jT' \neq O$ for any j < s. Suppose on the contrary that $n^jT' = O$, then we have $n^jT \in \langle P \rangle = \langle n^{r-1}S \rangle$. This contradicts the assumption that S and T are algebraically independent. We next show that S' and T' are algebraically independent. Suppose on the contrary that there is a non-trivial relation

$$n^m(an^{r-1-s}S'+bT') = O, \quad (\mathrm{GCD}(a,n) = 1, \ \mathrm{GCD}(b,n) = 1, \ 0 \leq m < s).$$

(Note that any non-trivial relation can be expressed as above since the orders of S' and T' are n^{r-1} and n^s , respectively.) Then we have

$$n^{m}\{an^{r-1-s}(\phi-1)(S) + b(\phi-1)(T)\} = O \Leftrightarrow (\phi-1)(n^{m}(an^{r-1-s}S + bT)) = O \Leftrightarrow n^{m}(an^{r-1-s}S + bT) \in \langle P \rangle = \langle n^{r-1}S \rangle.$$

Therefore, there is some $c \in \mathbb{Z}/n\mathbb{Z}$ such that $n^m(an^{r-1-s}S+bT)=cn^{r-1}S$. The multiplication by n^{s-m} on the both sides of the above equation induces $an^{r-1}S=O$, which is a contradiction since the order of S is n^r and GCD(a,n)=1.

Assume that the statement of the lemma is true for i-2 and i-1. We first show that the orders of $f^{(i)}(S)$ and $f^{(i)}(T)$ are n^{r-i} and n^s , respectively. From the induction hypothesis that $f^{(i-1)}(S)$ has order n^{r-i+1} and that r-i+1>s, we can represent it in the form

$$f^{(i-1)}(S) = an^{i-1}S + bT$$
, $(GCD(a, n) = 1, b \in Z/n^sZ)$.

Therefore, we have

$$\begin{split} n^{r-i}f^{(i)}(S) &= n^{r-i}(\phi-1)f^{(i-1)}(S) \\ &= n^{r-i}(\phi-1)(an^{i-1}S+bT) \\ &= (\phi-1)(an^{r-1}S+bn^{r-i}T) \\ &= (\phi-1)(an^{r-1}S) = O. \end{split}$$

(Note that $n^{r-i}T = O$ since we now consider the case $i \le r - s \Leftrightarrow s \le r - i$.) Also, if $n^j f^{(i)}(S) = O$, then

$$(\phi - 1)(n^j f^{(i-1)}(S)) = O \Leftrightarrow n^j f^{(i-1)}(S) \in \langle P \rangle = \langle n^{r-1} S \rangle.$$

It follows that $j+1 \ge r-i+1 \Leftrightarrow j \ge r-i$ since $n^{j+1}f^{(i-1)}(S) = O$. Thus, the order of $f^{(i)}(S)$ is n^{r-i} . On the other hand, from the induction hypothesis that the

order of $f^{(i-1)}(T)$ is n^s , it follows that $n^s f^{(i)}(T) = (\phi - 1)(n^s f^{(i-1)}(T)) = O$. To prove that the order of $f^{(i)}(T)$ is n^s , it is sufficient to show that $n^j f^{(i)}(T) \neq O$ for any j < s. Suppose on the contrary that $n^j f^{(i)}(T) = O$ for some j < s, then we have

$$n^{j}(\phi - 1)f^{(i-1)}(T) = O \Leftrightarrow n^{j}f^{(i-1)}(T) \in \langle P \rangle = \langle n^{r-1}S \rangle.$$

Therefore, $n^{j+1}f^{(i-1)}(T) = O$, from which it follows that $j+1 \geq s$. Thus, we obtain j = s-1 (note that j < s), and furthermore, $n^{s-1}f^{(i-1)}(T) = an^{r-1}S$ for some $a \in (Z/nZ)^{\times}$. Then we have

$$n^{s-1}f^{(i-1)}(T) = an^{r-1}S \Leftrightarrow (\phi - 1)(n^{s-1}f^{(i-2)}(T)) = an^{r-1}S,$$

and it follows that

$$\phi(n^{s-1}f^{(i-2)}(T)) = n^{s-1}f^{(i-2)}(T) + an^{r-1}S.$$
(2)

Thus we have

$$\begin{split} O &= n^{s-1} f^{(i)}(T) \\ &= n^{s-1} (\phi - 1)^2 (f^{(i-2)}(T)) \\ &= n^{s-1} (\phi^2 - 2\phi + 1) (f^{(i-2)}(T)) \\ &= n^{s-1} \{ (t-2)\phi (f^{(i-2)}(T)) + (1-q)f^{(i-2)}(T) \} \quad \text{(since } \phi^2 = t\phi - q) \\ &= (t-2) \{ n^{s-1} f^{(i-2)}(T) + a n^{r-1} S \} + (1-q) (n^{s-1} f^{(i-2)}(T)) \quad \text{(since } (2)) \\ &= (t-1-q) (n^{s-1} f^{(i-2)}(T)) + (t-2) a n^{r-1} S \\ &= (t-2) a n^{r-1} S, \end{split}$$

where the last equality follows from the assumption that $n|\#E(F_q) = 1 + q - t$ and that the order of $f^{(i-2)}(T)$ is n^s . Thus, we obtain $t-2 \equiv 0 \mod n$ since $a \in (Z/nZ)^{\times}$. Therefore, it follows that $q-1 \equiv 0 \mod n$. This contradicts the assumption that n/q-1.

We next show that $f^{(i)}(S)$ and $f^{(i)}(T)$ are algebraically independent. (This is proved similarly as in the case i = 1.) Suppose on the contrary that there is a non-trivial relation

$$n^{m} \{an^{r-i-s}f^{(i)}(S) + bf^{(i)}(T)\} = O$$

$$(GCD(a, n) = 1, GCD(b, n) = 1, 0 < m < s).$$

(Note that the orders of $f^{(i)}(S)$ and $f^{(i)}(T)$ are n^{r-i} and n^s , respectively, and that $r-i \geq s$.) Then we have

$$\begin{split} n^m(\phi-1)(an^{r-i-s}f^{(i-1)}(S)+bf^{(i-1)}(T)) &= O \\ \Leftrightarrow n^m(an^{r-i-s}f^{(i-1)}(S)+bf^{(i-1)}(T)) &\in \langle P \rangle = \langle n^{r-1}S \rangle. \end{split}$$

Therefore, the multiplication by n^{s-m} on the above last formula induces

$$n^{s} \{ an^{r-i-s} f^{(i-1)}(S) + bf^{(i-1)}(T) \} = O$$

 $\Leftrightarrow an^{r-i} f^{(i-1)}(S) = O,$

which is a contradiction since $f^{(i-1)}(S)$ has order n^{r-i+1} and $\mathrm{GCD}(a,n)=1$. The proof is completed. \square

Speeding up the Discrete Log Computation on Curves with Automorphisms

I. Duursma¹, P. Gaudry², and F. Morain² * **

¹ Université de Limoges, Laboratoire d'Arithmétique Calcul formel et Optimisation, 123 avenue Albert Thomas, F-87060 Limoges CEDEX, France duursma@unilim.fr

² Laboratoire d'Informatique de l'École polytechnique (LIX)
F-91128 Palaiseau CEDEX, France
{gaudry, morain}@lix.polytechnique.fr
http://www.lix.polytechnique.fr/

Abstract. We show how to speed up the discrete log computations on curves having automorphisms of large order, thus generalizing the attacks on anomalous binary elliptic curves. This includes the first known attack on most of the hyperelliptic curves described in the literature.

Keywords: elliptic and hyperelliptic curves, discrete logarithm, automorphisms.

1 Introduction

The use of elliptic curves in cryptography was first suggested by Miller [28] and Koblitz [19], following the work of Lenstra on integer factorization [23]. Many people improved on these ideas and the domain is flourishing. (See for instance the many books on that topic, e.g. [18,27].) Koblitz [20] was the first to suggest using hyperelliptic curves for the same goal.

The security of many cryptosystems based on curves relies on the difficulty of the discrete log problem. In some special cases, it was shown that this problem was rather easy [26], [11], [44], [38], [40]. Apart from new lifting ideas [43,17,7,6,15] that remain to be tested, it seems that the discrete log on elliptic curves still resists. On ordinary curves, the only known attack is a parallelized version of Pollard's rho method [50]. The Certicom challenge¹ records the state of the art in the field.

Among the curves suggested for cryptographic use, the so-called Anomalous Binary Curves (ABC curves) have been shown to be somewhat less secure than

^{*} On leave from the French Department of Defense, Délégation Générale pour l'Armement.

^{**} This work was supported by Action COURBES of INRIA (action coopérative de la direction scientifique de l'INRIA).

See http://www.certicom.com/chal/.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 103–121, 1999. © Springer-Verlag Berlin Heidelberg 1999

ordinary curves, due to the existence of an automorphism of large order. Two types of attack have been suggested [52], [13]. The first one can be seen as an additive rho method, the second one as a multiplicative method.

The purpose of this article is to point out that these attacks can be generalized to the case where there exists an automorphim on the curve (resp. on the Jacobian of an hyperelliptic curve). In particular, we show how to obtain a speedup of \sqrt{m} if there is an automorphism of order m.

The organization of the article is as follows. First of all, we recall Pollard's algorithm and its additive and multiplicative variants. Then, we describe the use of equivalence classes as originated in [52,13]. In particular, we show how to tackle the problem of useless cycles in the Wiener-Zuccherato approach, by analyzing the number of useless cycles and being able to throw them out. We then show how to use automorphisms of large order on (hyper)elliptic curves, including generalized ABC curves [30,45] and curves with CM by $\mathbb{Z}[\sqrt{-1}]$ or $\mathbb{Z}[(1+\sqrt{-3})/2]$, for which this appears to be the first published attack. This method applies also to hyperelliptic curves that were presented by Koblitz and others. We support our approach by numerical simulations.

2 Parallel Collision Search

Let G be a finite abelian group with law written additively \oplus (and multiplication by k denoted by [k]P) and P an element of order n of G (w.l.o.g. we will assume that n is prime using the Pohlig-Hellman approach [33]). Let Q be an element of P, the cyclic group generated by P. The discrete log problem refers to the search for R s.t. Q = [R]P. For instance, R can be the group of points of an elliptic curve over a finite field, or the Jacobian of a hyperelliptic curve.

Generically, there exist three methods for computing discrete logs in time $O(\sqrt{n})$: Shanks's method [41], Pollard's rho and lambda methods [34], the last one being better when one has to find the discrete logarithm in a reduced range (that is not over [0, n[)). The first method also uses $O(\sqrt{n})$ space, which is generally a problem. After recalling Pollard's rho, we show how a parallel collision search method can be described in the same spirit.

2.1 Pollard's Rho Method

It is well known that if we collect approximately k random values from a set of cardinality n, then we will find two equal values if $k = \sqrt{\pi n/2}$ on average. To solve the memory problem, Pollard [34] suggested to iterate a random function f on the group $\langle P \rangle$. Starting from a point $R_0 \in \langle P \rangle$, for instance $R_0 = [u_0]P \oplus [v_0]Q$ with u_0 and v_0 random integers (modulo n), one computes

$$R_{i+1} = f(R_i)$$

thereby obtaining two sequences (u_i) and (v_i) modulo n such that:

$$R_i = [u_i]P \oplus [v_i]Q.$$

In order to understand the properties of this random sequence, one is led to study the functional graph of f, that is the graph built using all starting points $R \in P$, as shown in Figure 1.

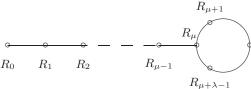


Fig. 1. The rho shape of (R_i) .

Such a sequence consists in a tail of length μ and a cycle of length λ , thus forming a rho of size $\rho = \mu + \lambda$. Asymptotic values for the critical parameters of this graph are given in [10]. For instance, a graph with n vertices should have $0.5 \log n$ (connected) components, the average tail (resp. cycle) length should be $\sqrt{\pi n/8}$, thus giving a rho length of about $\sqrt{\pi n/2}$.

These results imply that after about $\sqrt{\pi n/2}$ iterations, we will find a collision, that is two integers i and j for which $R_i = R_j$. This will give an identity

$$[u_i - u_j]P \oplus [v_i - v_j]Q = 0$$

from which we can deduce κ if $v_i \not\equiv v_j \mod n$, which happens with probability 1 - 1/n: Indeed, each element R of < P > has n different representations as $[u]P \oplus [v]Q$ and the values v_i and v_j are supposed to be random integers in [0, n-1].

2.2 Using Distinguished Points

An efficient implementation of parallel collision search requires storing a limited number of points that have a distinguished property (this idea was used for the first time in [35]). If θ is the proportion of distinguished points, then we should find a collision after computing about $\sqrt{\pi n/2}$ points, or $\theta \sqrt{\pi n/2}$ distinguished points. Now, parallelization is straightforward: have each processor contribute to the same list of distinguished points [50]. Each processor would have to find $\theta \sqrt{\pi n/2}/M$ points, thus yielding a speedup of M to the total running time.

To understand how a parallelized search works, consider figure 2, on which we have drawn two paths, the one corresponding to processor i and the second to processor j. Their paths collide at point C, which lies between the distinguished points D_1 and D_3 (resp. D_2 and D_3). The collision will be discovered as soon as we find that $R = D_3$.

Such a random walk has a very important feature: it is *deterministic*, which means that if f(R) = C, then f(f(R)) = f(C), so that after hitting C from any direction, one follows a single path afterwards.

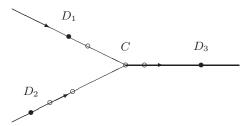


Fig. 2. The deterministic property.

2.3 Choosing the Function f

An Additive Random Walk. One way of choosing a good random function consists in precomputing r random points $(T^{(j)})_{0 \le j < r}$ in P > 0, with $T^{(j)} = [u^{(j)}]P \oplus [v^{(j)}]Q$ and to define the random walk as

$$f(R) = R \oplus T^{(j)}$$

where $j = \mathcal{H}(R)$ with \mathcal{H} a hash function sending $\langle P \rangle$ to $\{0, 1, \dots, r-1\}$. This creates an *additive* random walk in $\mathbb{Z}/n\mathbb{Z}$ since each point R_i can be written as $R_i = [u_i]P \oplus [v_i]Q$ for which

$$u_{i+1} \equiv u_i + u^{(j)} \mod n, \quad v_{i+1} \equiv v_i + v^{(j)} \mod n.$$

Satler and Schnorr [39] have shown that the above approach is sufficiently random if $r \geq 8$. Teske has found experimentally [49] that a value of $r \geq 20$ is more convenient.

A Multiplicative Random Walk. One can also use a function f built with fixed multipliers $(\mu_j)_{0 \le j \le r}$, μ_j defined modulo n. We define:

$$f(R) = [\mu_j]R$$

where $j = \mathcal{H}(R)$ as above. In this version, we would in fact compute:

$$f(R) = \left[\prod_{j=0}^{r-1} \mu_j^{f_j}\right] R_0.$$

A collision with a single sequence would not be interesting since this would lead to $(\prod_j \mu_j^{f_j}) \equiv 1 \mod n$. A collision is interesting if it comes from two distinct sequences: two initial values $R_0 = [u_0]P \oplus [v_0]Q$ and $R'_0 = [u'_0]P \oplus [v'_0]Q$ would lead to a more useful $[\prod_j \mu_j^{f_j}]([u_0]P \oplus [v_0]Q) = [u'_0]P \oplus [v'_0]Q$. For this to be random enough, we must have r large as above. To be efficient, the method requires that the multiples be efficiently computed.

Mixed Walks. In practice, it would be fruitful to mix both strategies. This would lead to a walk that is difficult to analyze in theory but which is likely to be "more random".

3 Random Mappings on Equivalence Classes

Let \sim be an equivalence relation on < P > for which there are n/m equivalence classes. We note \overline{R} for the equivalence class of R. If we can iterate Pollard's rho on the set of equivalence classes $< P > / \sim$, then we should find a collision in time $\sqrt{\pi n/(2m)}$ instead of $\sqrt{\pi n/2}$.

The easiest way of building an equivalence class is by using automorphisms of the group G. Wiener and Zuccherato [52] considered the case where G is the set of points of an elliptic curve E defined over a finite field \mathbb{K} . In that case, the equivalence relation would be $S \sim T$ if and only if T = -S. The first known nontrivial example of such a phenomenon came from ABC curves [52,13] (see section 3.3 below).

More generally, if we know an automorphism α of order m operating on G, we can use the equivalence relation:

$$S \sim T \iff \exists i, S = [\alpha^i]T$$

and hope to get a speedup of \sqrt{m} . We will give new examples in the following section.

¿From now on, we suppose our equivalence class is built on an automorphism α of order m. For the method to work efficiently, the equivalence class of a point should be easy to compute, that is much faster than an addition in G.

3.1 Well Defined Mappings

The first thing we can think of is to iterate f as usual, but perform the matches on the equivalence classes only. Starting from a random point R in $\langle P \rangle$, we iterate R = f(R); if $\overline{R} = \overline{R_j}$ for some j, try to find a match and stop, otherwise, store \overline{R} .

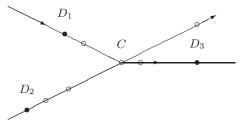


Fig. 3. Paths that cross each other.

This looks fine, as long as we do not want to parallelize the algorithm. In that case, we can come across the diagram of figure 3 which shows that the random walk would no longer be deterministic. For instance, imagine that we are again in the case of elliptic curves with an additive walk and that:

$$R_{i+1} = R_i + T^{(k)} = (x, y),$$

$$S_{j+1} = S_j + T^{(\ell)} = (x, -y).$$

In that case $\overline{S_{j+1}} = \overline{R_{i+1}}$, but in general $f(S_{j+1}) \neq f(R_{i+1})$.

Therefore, we must find a function f defined over P > and that is well defined over the equivalence classes. More precisely, we want that if $R' \in \overline{R}$, then $\overline{f(R')} = \overline{f(R)}$. The most obvious approach is to use $R_{i+1} = \overline{f(R_i)}$ and we iterate R = f(R) until a collision is found.

3.2 Useless Cycles for Additive Walks

Let us explain why useless cycles appear. Suppose that G is the set $E(\mathbb{K})$ of points on an elliptic curve E over the finite field $\mathbb{K} = \mathbb{F}_{p^n}$. Suppose we use a function f as in section 2.3. Assume that the representative of a class is the point (x,y) with least y. Then we could encounter a situation where starting from R_i , we find

$$R_{i+1} = \overline{R_i \oplus T^{(j)}} = \ominus R_i \ominus T^{(j)}.$$

It could happen that

$$R_{i+2} = \overline{R_{i+1} \oplus T^{(j)}}$$

(the same j), thus yielding

$$R_{i+2} = \overline{\ominus R_i \ominus T^{(j)} \oplus T^{(j)}} = R_i.$$

This proves the existence of useless 2-cycles.

We fix some notations for studying these cycles. We consider a (primitive) t-cycle consisting of points $R_1, R_2, \ldots, R_t, R_{t+1} = R_1$. We denote by j_k the value $\mathcal{H}(R_k)$ of the hash function and we denote by $\varepsilon_k = \alpha^{e_k}$ the automorphism which gives the representative of the equivalence class of $\overline{R_k \oplus T^{(j_k)}}$. Thus we have $R_{k+1} = [\varepsilon_k](R_k \oplus T^{(j_k)})$. See figure 4 for a description of the cycle.

We can now derive an expression of R_{t+1} in terms of R_1 and our parameters j_k and ε_k :

$$[1/\varepsilon_t]R_{t+1} = [\varepsilon_{t-1}\cdots\varepsilon_1]R_1 \oplus \tau$$

with $\tau = [\varepsilon_{t-1} \cdots \varepsilon_1] T^{(j_1)} \oplus [\varepsilon_{t-1} \cdots \varepsilon_2] T^{(j_2)} \oplus \cdots \oplus [\varepsilon_{t-1}] T^{(j_{t-1})} \oplus T^{(j_t)}$. We have a useless t-cycle if $\tau = 0$. Indeed, when $\varepsilon_1, \ldots, \varepsilon_{t-1}$ and j_1, \ldots, j_t are chosen such that $\tau = 0$, then R_{t+1} is in the class of R_1 , which does not impose any restriction on ε_t . The fact that the $T^{(j)}$ are randomly chosen implies that (very likely) there is no simple relation between them². In that case, a necessary

² If there is one, then we have found the discrete log!

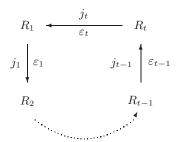


Fig. 4. A typical useless t-cycle.

condition for having $\tau = 0$ is that for all $k \in \{1, 2, ..., t\}$ there exists a $l \neq k$ such that $j_k = j_l$.

Now, we can summarize the following results on the expected number of useless t-cycles.

Proposition 31 Suppose there is an automorphism α of order m acting on a group G with n elements and that the $T^{(j)}$ are not related by relations involving powers of α . We can estimate the probability $\mathcal{P}(t)$ of useless t-cycles for small t when iterating an additive function with r branches (the expected number of t-cycles being $\mathcal{P}(t)(n/m/t)$).

We have

$$\mathcal{P}(t) \leq \sum_{k=1}^{\min(r,t/2)} \frac{1}{m^k} \min\left(1, \frac{r!k^t}{(r-k)!r^t}\right).$$

Moreover, for small values of t we have the following bounds:

t	pattern	$\mathcal{P}(t)$	
2	$j_1 = j_2$	1/(mr)	
3		$\leq 2/(mr)^2$	$3 \mid m$
Ш		0	$3 \nmid m$
4	$j_1 = j_2 = j_3 = j_4$	$\leq (1 - 1/m)^2/(mr^3)$	
Ш	$j_1 = j_3, j_2 = j_4$	$(1-1/r)/(mr)^2$	

The proof of this proposition is given as an appendix. This proposition shows that when t and r are large enough, the probability to find a t-cycle is very small, and we likely do not encounter one.

Taking Care of Useless Cycles. Using Proposition 31, it is clear that an easy way of solving the problem is to force a large value of r, which is not really a problem in practice. In the rare cases where a cycle appears, we can get out of it by collapsing it as suggested in [13]. If the cycle is $R_1 \mapsto R_2 \mapsto \cdots \mapsto R_t$, we want to get out of it in a symmetric way, that is reach the same point, whichever R_i was the point at which we entered the cycle. Our version is to sort the points R_i to obtain S_1, S_2, \ldots, S_t and start again, say, from $R = \bigoplus_{i=1}^t [i^i + 1]S_i$. Anything that breaks linearity would be convenient.

From a practical point of view, we count the number of distinguished points we obtain along the path. If this number does not evolve as prescribed by the theory, that is 1 among $1/\theta$ points, then we decide to inspect the cycle and check out if it comes from a useless one or not. The more frequent case of 2-cycles can be handled with a look-ahead technique (see [52]) that is somewhat simpler.

3.3 A Multiplicative Random Walk

Let us explain the situation for ABC curves [22]. In this case, $\mathbb{K} = \mathbb{F}_{2\ell}$ and E is defined over \mathbb{F}_2 . Then $\lambda: (x,y) \mapsto (x^2,y^2)$ is an automorphism of order ℓ . If ℓ is odd³, the equivalence relation is: $S \sim T \iff \exists i, S = [\pm \lambda^i]T$ with $m = 2\ell$ classes. This relation has been exploited in [13] and [52] as well as in real life computations by R. Harley (see http://www.certicom.com/chal/).

Gallant et al. have suggested to use a multiplicative random walk with a function f defined by $f(R) = [\mu_{\overline{R}}]R$, where $\mu_{\overline{R}}$ is a multiplier depending on the class of R. This walk is well defined, since f commutes with λ :

$$f(R') = [\mu_{\overline{R'}}]R' = [\mu_{\overline{R}}]R' = [\mu_{\overline{R}}]([\pm \lambda^i]R) = [\pm \lambda^i]([\mu_{\overline{R}}]R) \in \overline{f(R)}.$$

In order to make the method efficient, the multiples must be computed very quickly. In that case, $\mu_{\overline{R}}$ is taken to be $1 + \lambda^{\mathcal{H}(\overline{R})}$ where \mathcal{H} is a hash-function sending the equivalence classes to $\{0, 1, \dots, m-1\}$. As noted in section 2.3, the number of such multipliers should be large. For the CM examples given below this will prove not to be the case.

4 Finding New Examples

Our idea is to find new examples of curves with non-trivial automorphisms. We will concentrate on algebraic curves defined over finite fields of characteristic p. We will first summarize the results concerning the number of automorphisms of (the Jacobian of) a curve.

4.1 Theoretical Results

For g = 1, we know that we can obtain automorphisms of order 4 or 6 for some CM curves (details for both cases follow in Section 4.2) and powers of the Frobenius for ABC curves [22].

We obtain examples for g > 1 by considering CM hyperelliptic curves and curves defined over finite fields fixed by powers of the Frobenius, as well as curves defined over \mathbb{Q} having non-trivial automorphisms.

For genus g > 1, we need a priori distinguish between automorphisms of a curve and automorphisms of its Jacobian. Automorphisms of a curve naturally define automorphisms on its Jacobian. By Torelli's theorem [29], all automorphisms of the Jacobian (for a fixed projective embedding and a chosen zero

³ If ℓ is even, we obtain a relation with ℓ classes only.

element) arise in this way (except for multiplication by minus one on the Jacobian of non-hyperelliptic curves). And we may identify the automorphisms of a curve and those of its Jacobian.

If g > 1, the number of geometric automorphisms (not considering powers of the Frobenius) does not exceed 84(g-1) provided that p > g+1, with the exception of the curve $y^2 = x^p - x$ (see [36]). Without the restriction p > g+1, the upper bound becomes $16g^4$, with again a single explicit exception: $y^q - y = x^{q+1}$ or a quotient thereof, for q a power of p (see [48]). Under the assumption that the Jacobian has maximal p-rank g, the upper bound 84g(g-1) holds [32].

The exceptions are of no interest to us as the Jacobians of these curves contain no large cyclic components. To some extent this occurs more generally for curves with many automorphisms. Automorphisms induce invariant subgroups in the Jacobian. When these subgroups are non-trivial for many different automorphisms, the Jacobian does not admit cyclic components of large prime order. For such a large component would itself have invariant subgroups. It follows that the selection of curves with automorphisms suitable for cryptosystems requires some care. For the decomposition of a Jacobian with given automorphism group, see [1,16].

Examples of curves with automorphisms that have large cyclic components in their Jacobian are the curves $y^2 = x^p - x + 1$ in odd characteristic p [9]. The automorphism $x \mapsto x + 1$ of prime order p induces as invariant subgroup the trivial group and the Jacobian itself is often of prime order. Other examples are described in Section 4.3. In this paper we will always consider that the automorphism on the Jacobian can be restricted to an automorphism of the cyclic subgroup we are working in. This will be the case for all practical examples where the group is almost prime. Indeed, assume that we are dealing with a cyclic subgroup of prime cardinality n generated by P. Then n^2 does not divide the cardinality of the whole group, and the only elements of order n in the group are precisely those which constitute the subgroup $\langle P \rangle$. Then the image of P by the automorphism is a of order n, and is in $\langle P \rangle$, and the subgroup is stable under the action of the automorphism.

4.2 New Examples with Elliptic Curves

We can find the proofs for what follows in [42, Chapter 2]. It is well known that the only elliptic curves having non-trivial automorphisms (over $\overline{\mathbb{Q}}$) have CM by $\mathbb{Z}[i]$ (resp. $\mathbb{Z}[\rho]$) where $i^2 = -1$ (resp. $\rho^3 = 1$). They are: $E_{a,0}: Y^2 = X^3 + aX$ for $a \neq 0$ and $E_{0,b}: Y^2 = X^3 + b$ with $b \neq 0$. On $E_{a,0}$, multiplication by i is an automorphism sending a point (x,y) to (-x,iy); on $E_{0,b}$, ρ sends (x,y) to $(\rho x, y)$.

Over \mathbb{Q} , this is all the story, an elliptic curve having in general automorphism group $\{\pm 1\}$. Over finite fields, Frobenius automorphisms enter the game as seen with ABC curves. It is easy to see how to generalize the automorphism attack to curves suggested by Müller [30] and Smart [45].

The reductions of the curves $E_{a,0}$ and $E_{0,b}$ are supersingular for p=2,3, so that we will consider them interesting only in the case p>3. For the sake of simplicity, we suppose that $\mathbb{K}=\mathbb{F}_p$, p odd prime >3.

The Case $E_{a,0}$. We suppose that $a \not\equiv 0 \bmod p$ and $p \equiv 1 \bmod 4$ (if $p \equiv 3 \bmod 4$, then $E_{a,0}$ is supersingular and the MOV reduction applies [26]). Again, we suppose that we are looking for a discrete log on the group < P > with prime order n. Let $I_p^2 \equiv -1 \bmod p$ and $I_n^2 = -1 \bmod n$ such that $[I_n](x,y) = (-x,I_py)$ (this is possible since p and n are n are n are n and n are n and n are n and n are n and the class of n curves). Then the automorphism is n and n are n and the class of n and the class of n are n are n and the class of n are n and n are n are n and n are n

The "obvious" choice would be to use a generalization of the ABC approach and use $\mu_{\overline{R}} = 1 + \alpha^{\mathcal{H}(\overline{R})}$ where \mathcal{H} sends the classes to $\{0,1,2,3\}$. However, this does not work, due to the fact that $\alpha^2 = -1$ and $(1 + \alpha)(1 - \alpha) = 2$. A random walk would compute multiples of the form $(1 + \alpha)^u(1 - \alpha)^v$ which is not random enough (compare with [39]). Using other multiples would be too costly.

So, we must use the additive random walk, with the modifications described in the preceding section. This gives us a speed up of $\sqrt{4} = 2$. In the case of $\mathbb{F}_{p^{\ell}}$ (ℓ odd), we can use the Frobenius $(x, y) \mapsto (x^p, y^p)$ to obtain a speedup of $\sqrt{4\ell}$.

The Case $E_{0,b}$. We suppose $b \not\equiv 0 \mod p$ and $p \equiv 1 \mod 3$ (the case $p \equiv 2 \mod 3$ yields a supersingular curve). We let ρ_p (resp. ρ_n) denote a primitive third root of unity modulo p (resp. modulo n) such that $[\rho_n](x,y) = (\rho_p x,y)$. The automorphism is $\alpha(x,y) = [-\rho_n](x,y)$ and the class of R = (x,y) consists of $\{(x,\pm y),(\rho_p x,\pm y),(\rho_p^2 x,\pm y)\}$. We take the representative with smallest x and smallest y.

Here again, the multiplicative choice using multipliers of the form $(1+\alpha^i)$ is disastrous due to the fact that $1+\alpha^2=-\alpha$ and $1+\alpha=-\alpha^2$ and we come back to the additive version, thus yielding a speedup of $\sqrt{6}$ and more generally $\sqrt{6\ell}$ over \mathbb{F}_{p^ℓ} , ℓ odd.

4.3 Examples of Hyperelliptic Curves

We will consider curves of equation $Y^2 = F(X) = X^{2g+1} + \cdots$ when p > 2 (resp. $Y^2 + H(X)Y = F(X) = X^{2g+1} + \cdots$ for p = 2) where g > 1 is the genus of the curve. There is no group law on the curve, but there is one on its Jacobian, noted Jac(C). The only result we need says that every element of Jac(C) can be uniquely represented by a so-called reduced divisor with at most g points. For more precise statements about those things, one can refer to Mumford [31]. In [4] (see also [20]), Cantor gives an algorithm for computing with the reduced divisors: we can add them in polynomial time. We do not recall his method here, but we give the representation of a reduced divisor: it is a pair of polynomials [u(z), v(z)], where u is monic of degree at most g and greater than

the degree of v. The opposite of such a reduced divisor (for the Jacobian group law) is just the divisor represented by [u(z), -v(z)]. The hyperelliptic involution is $[u(z), v(z)] \mapsto [u(z), -v(z)]$.

From a practical point of view, it is not as easy to compute the cardinality of a Jacobian over finite fields as it is for elliptic curves using the so-called SEA algorithm [25,24]. So various researchers have suggested some special form of curves for which this computation is easy [21], [9], [5], [46], [37], [3].

Automorphisms. We suppose α is an automorphism on the curve of the form $\alpha:(x,y)\mapsto(\alpha_1(x),\alpha_2(y))$. We restrict ourself to the study of two classes of automorphisms: in case 1, the coordinate-functions α_1 and α_2 are identical and are also automorphisms (a typical example is the Frobenius action), and in case 2, α_1 and α_2 are multiplication maps in the finite field (this occurs for CM curves).

In both cases the automorphism can be extended to the Jacobian of the curve, and we still denote α the extended automorphism. With the polynomial representation of reduced divisors, we can derive simple expressions for the action of α : for $h \leq g$, we have

$$\alpha : [z^{h} + u_{h-1}z^{h-1} + \dots + u_{0}, v_{h-1}z^{h-1} + \dots + v_{0}]$$

$$\mapsto [z^{h} + \alpha_{1}(u_{h-1})z^{h-1} + \dots + \alpha_{1}(u_{0}), \alpha_{1}(v_{h-1})z^{h-1} + \dots + \alpha_{1}(v_{0})]$$

for case 1, and

$$\alpha : [z^{h} + u_{h-1}z^{h-1} + \dots + u_{0}, v_{h-1}z^{h-1} + \dots + v_{0}]$$

$$\mapsto [z^{h} + \alpha_{1}u_{h-1}z^{h-1} + \dots + \alpha_{1}^{h}u_{0}, \alpha_{2}\alpha_{1}^{-(h-1)}v_{h-1}z^{h-1} + \dots + \alpha_{2}v_{0}]$$

for case 2.

Thus, provided that the computation of the action of α_1 and α_2 is easy we can obtain the image of a divisor quite quickly.

Examples. In the literature we find many examples of hyperelliptic curves which come with an automorphism. Some of them are obvious (such as Frobenius endomorphism), but some others were probably not known to the authors (see the curve of Sakai and Sakurai [37]). We summarize in table 1 some examples for which we give the non-trivial automorphisms, and the order m that we obtain when combining them together with the hyperelliptic involution.

The curves marked by (*) can be broken by the reduction method of [11] which is much faster than the ρ method, even with the use of automorphisms. For high genus curves in the examples marked by (\dagger) , a faster attack is given by the index-calculus method of [2].

For each curve in the table, we can improve the parallel collision search by a factor \sqrt{m} . For example, if one wants to break the cryptosystem proposed by Sakai and Sakurai, which uses a divisor of prime order around 2^{171} , we have to perform about 2^{86} operations on the Jacobian, and using the Frobenius, the

Author	Equation of curve	Field	Automorphisms	m
Koblitz [20], [21] $Y^2 + Y = X^5 + X^3$ (*)		\mathbb{F}_{2^n}	Frob + $\begin{cases} X \mapsto X + 1 \\ Y \mapsto Y + X^2 \end{cases}$	4n
	$Y^2 + Y = X^5 + X^3 + X \ (*)$	\mathbb{F}_{2^n}	Frobenius	2n
	$Y^2 + Y = X^{2g+1} + X$	\mathbb{F}_{2^n}	Frobenius	2n
	$Y^2 + Y = X^{2g+1}$	\mathbb{F}_{2^n}	Frobenius	2n
Buhler Koblitz [3]	$Y^2 + Y = X^{2g+1} \ (\dagger)$	\mathbb{F}_p with		2(2g+1)
Chao et al. [5]		$p \equiv 1 (2g+1)$		
Sakai Sakurai [37]	$Y^2 + Y = X^{13} + X^{11} +$	$\mathbb{F}_{2^{29}}$	Frobenius &	4×29
			$X \mapsto X + 1$	
	$X^9 + X^5 + 1 (\dagger)$		$Y \mapsto Y + X^6 + X^5$	
			$+X^4+X^3+X^2$	
Duursma &	$Y^2 = X^p - X + 1 \ (\dagger)$	\mathbb{F}_{p^n}	Frobenius &	2np
Sakurai [9]			$(X,Y) \mapsto (X+1,Y)$	

Table 1. Examples of curves

involution and the new automorphism, we have only to perform about 2^{82} operations. Note that the authors took into account the use of Frobenius in their evaluation of the security, but not the other automorphism (however it still does not break the system).

The Curve $Y^2 = X^5 - 1$ and Generalizations. The Jacobian of the curve $Y^2 = X^5 - 1$ admits complex multiplication by the field $\mathbb{Q}(\zeta_5)$ where ζ_5 is a 5-th root of unity. Combining it with the hyperelliptic involution, we obtain an automorphism of order 10. The formulae for the action on reduced divisors are:

This case may be generalized to the curves of equations $Y^2 = X^{2g+1} - 1$ (or $Y^2 + Y = X^{2g+1}$ as suggested in [3]), which admit complex multiplication by the ring of integers of $\mathbb{Q}(\zeta_{2g+1})$, providing an automorphism of order 2g+1 (and even 4g+2 combined with the trivial involution).

In this case, we could also dream of using a multiplicative random walk with multipliers $1+\alpha^i$. However, we have to be careful, due to the many algebraic relations between 5-th roots of unity. Apart from $1-\alpha+\alpha^2-\alpha^3+\alpha^4=0$, we have among others $1+\alpha^5=0$, $(1+\alpha^2)(1+\alpha^4)=\alpha^3$, $(1+\alpha^4)(1+\alpha^8)=\alpha$, $(1+\alpha^6)(1+\alpha^8)=-\alpha^2$, $(1+\alpha^3)(1+\alpha^8)=1+\alpha$, etc. If during the random walk we do a couple of consecutive steps corresponding to one of these relations, then we obtain a cycle of length 2 which is useless. So, we come back to the additive one to get our speedup of $\sqrt{10}$.

5 Numerical Experimentations

In order to validate our findings, we made several experiments on random graphs of reasonable size as well as some real discrete log computations on small examples. The hash function used was $\mathcal{H}((x,y)) = y \mod r$ in each case. All examples were done using the computer algebra system MAGMA.

5.1 Elliptic Examples

We built functional graphs for the case of $E:Y^2=X^3+2X$ over $\mathbb{K}=\mathbb{F}_{1000037}$. We chose P=(301864,331917) of prime order n=500153. We made two series of experiments on 50 random mappings with r branches. The following table contains the number of useless t-cycles found:

r	#2 - cycles	#4 - cycles
20	772(781)	5(5)
1000	16(15)	0(0)

These values are close to those prescribed by Proposition 31 that are given in parentheses.

We did the same for the curve $E: Y^2 = X^3 + 2$ over $\mathbb{K} = \mathbb{F}_{1000381}$ on which P = (1,696906) is a point of prime order n = 998839. Again we tried 50 random walks.

r	#2 - cycles	#4 - cycles	#3 - cycles
20	694(693)	7(7)	2(3)
1000	13(14)	0(0)	0(0)

We also did 100 real discrete log computations for the curves given below. We give the average gain obtained, that is the ratio $\theta\sqrt{\pi n/2}$ divided by the number of distinguished points computed. For $E:Y^2=X^3+2X$ over $\mathbb{F}_{10000003021}$ with P=(9166669436,170163551) of prime order n=5000068261, Q=(1314815213,7654067643) (with $\kappa=2153613198$), we got a speed up of 1.88 for r=20 (resp. 1.82 for r=1000) and for $E:Y^2=X^3+2$ over $\mathbb{F}_{10000000963}$, P=(2,5825971627), n=9999804109, Q=(4,6715313768) (thus $\kappa=8959085671$), we got 2.23 for r=1000.

5.2 Hyperelliptic Examples

We built the functional graphs of 50 random mappings for the Jacobian of the curve $Y^2 = X^5 - 1$ over $\mathbb{K} = \mathbb{F}_{31^3}$, which has a divisor of prime order n = 778201. We found 201 components on average, with 196 2-cycles and 0.8 4-cycles, which is closed to the expected theoretical values (194 and 0.7).

We did the same for the first example given by Koblitz in [20]: the curve is $Y^2 + Y = X^5 + X^3 + X$ over $\mathbb{K} = \mathbb{F}_{2^{11}}$, with a divisor of order n = 599479. We tried 50 different pseudo-random walks obtaining on average 31.5 2-cycles (the theory predicts 30.9), and almost no 4-cycles (0.07 expected). With the modified walk, we have 5.6 components on average (5.1 for pure random walk).

We also did some experiments of discrete log computations. For each of the curves, we did 1000 tests, the r parameter was fixed to 50. The first curve was $Y^2 = X^5 - 1$ over \mathbb{F}_{31^7} , with a divisor of order n = 1440181261. We got an average gain of 3.13 for the number of iterations (to be compared with $\sqrt{10} \approx 3.16$). The

second test was done for the curve $Y^2+Y=X^5+X^3+X$ over $\mathbb{F}_{2^{37}}$, with a divisor of order n=319020217. The average gain obtained was 8.83 (theory says $\sqrt{74}\approx 8.60$).

6 Conclusions

We have described a general framework speeding up discrete logarithm computations on curves with large automorphism groups, including in particular some elliptic and hyperelliptic curves with complex multiplication. One could raise the question to find curves having automorphisms which are not of Frobenius or CM type.

Note that for high genus hyperelliptic curves, we can speed up the Adleman–DeMarrais–Huang discrete log algorithm [2] using all these automorphisms (see [14]). This suggests that the use of such curves in cryptosystems requires great care.

For the genus > 2, most of the curves in the literature for which the cardinality of the Jacobian can be computed have a non-trivial automorphism. It would be interesting to build new curves having no automorphisms. For genus 2, examples are given by the CM construction of Spallek [46], see also [51]. See also [47] for some examples of high genus random curves and [12] for superelliptic curves.

Acknowledgments. We are grateful to Projet CODES at INRIA, and particularly Daniel Augot, for having given the authors the opportunity to meet and work together. We also to thank E. Thomé for helpful discussions concerning this work, as well as R. Harley for his careful reading of an intermediate version of the article. Also, we are very happy to receive constructive reports from the referees, a phenomemon which is so rare that we want to emphasize it.

A Proof of Proposition 31

Remember that α is an automorphism of order m and that we use an additive function with r branches.

A.1 2-cycles

How many useless 2-cycles do we expect? We first evaluate the probability for a random point R_1 to be in a useless 2-cycle. Let R_2 be the point computed by the pseudo-random walk, and $j_1 = \mathcal{H}(R_1)$. We have $R_2 = \overline{R_1 \oplus T^{(j_1)}}$. The useless cycle can be produced when the class of $R_1 \oplus T^{(j_1)}$ is $\ominus R_1 \ominus T^{(j_1)}$, which occurs with probability 1/m. If this first condition is satisfied, the cycle is produced when $\mathcal{H}(R_2)$ equals j_1 , which occurs with probability 1/r. Finally, the probability for a point to be in a useless 2-cycle is 1/(rm).

A.2 3-cycles

For 3-cycles, the expression of τ is

$$\tau = [\varepsilon_1 \varepsilon_2 R_1 + \varepsilon_1 \varepsilon_2] T^{(j_1)} \oplus [\varepsilon_2] T^{(j_2)} \oplus T^{(j_3)}.$$

The condition on the j_k 's implies that $j_1 = j_2 = j_3$. Then $\tau = 0 = [\varepsilon_1 \varepsilon_2 + \varepsilon_2 + 1]T^{(j_1)}$, and we have $\varepsilon_2(1 + \varepsilon_1) = -1$. We study "true" 3-cycles, so we do not want $R_3 = R_1$, so we can suppose $\varepsilon_1 \neq -1$. Then $\varepsilon_2 = -1/(1 + \varepsilon_1)$.

If we suppose that the automorphism α is a root of unity⁴, the last equation gives $|1 + \varepsilon_1| = |\varepsilon_1| = 1$, and then ε_1 is a third root of unity. Thus, if the order m of α is not a multiple of 3, we cannot have a useless 3-cycle; otherwise we can give an upper bound for the probability for a point to be in a useless 3-cycle by $2/(mr)^2$ (ε_1 can take 2 values, and then ε_2 is unique).

A.3 4-cycles

We have three different patterns for the j_k 's.

First case: $j_1 = j_2 = j_3 = j_4$. Then we have $\tau = [\varepsilon_1 \varepsilon_2 \varepsilon_3 + \varepsilon_2 \varepsilon_3 + \varepsilon_3 + 1] T^{(j_1)}$, and we have a cycle if

$$\varepsilon_3 = -\frac{1}{1 + \varepsilon_2(1 + \varepsilon_1)}.$$

The properties $\varepsilon_1 \neq -1$ and $\varepsilon_2 \neq -1/(1+\varepsilon_1)$ are necessary to have a true 4-cycle, and guarantee that the expression makes sense. However, for some values of ε_1 and ε_2 , there is no ε_3 satisfying this equation. For example, if $\alpha^4 = 1$, then the only triples of solutions are $(\varepsilon_1, \varepsilon_2, \varepsilon_3) = (\alpha, \alpha, \alpha)$ or $(-\alpha, -\alpha, -\alpha)$.

We can bound the probability of a 4-cycle with this pattern. The probability that the j_k 's are equal is $1/r^3$; the probability that $\varepsilon_1 \neq -1$ is 1 - 1/m, the probability that $\varepsilon_2 \neq -1$ and $\varepsilon_2 \neq -1/(1 + \varepsilon_1)$ is bounded by (1 - 1/m), and finally we have at most one choice for ε_3 , i.e. probability 1/m. Finally, the probability for a point to belong to such a 4-cycle is less than $(1 - 1/m)^2/mr^3$.

Second case: $j_1 = j_2$ and $j_3 = j_4$. We want to have a true 4-cycle, so $R_1 \neq R_3$, which implies that $\varepsilon_1 \neq -1$. In that case, the equation $\tau = 0$ becomes

$$[\varepsilon_2\varepsilon_3(1+\varepsilon_1)]T^{(j_1)}\oplus[1+\varepsilon_3]T^{(j_3)}=0,$$

with $T^{(j_1)} \neq T^{(j_3)}$, so this equation has no solution with $\varepsilon_1 \neq -1$ if the points $T^{(j)}$ are sufficiently randomly chosen. Hence such a pattern cannot occur with significant probability.

The same result holds for the case: $j_1 = j_4$ and $j_2 = j_3$.

Third case: $j_1 = j_3$ and $j_2 = j_4$. In that case, the equation $\tau = 0$ becomes $[\varepsilon_3(\varepsilon_1\varepsilon_2 + 1)]T^{(j_1)} \oplus [\varepsilon_2\varepsilon_3 + 1]T^{(j_2)} = 0$, which reduces to the system

$$\begin{cases} \varepsilon_1 \varepsilon_2 = -1, \\ \varepsilon_2 \varepsilon_3 = -1, \end{cases}$$

 $^{^{4}}$ This is always the case for an automorphism of a cyclic group.

or $\varepsilon_3 = \varepsilon_1 = -1/\varepsilon_2$. If we assume that the automorphism is a root of unity, then for any given ε_1 , there is a unique solution of the system for ε_2 and ε_3 .

The probability that $j_1 = j_3$ is 1/r, idem for $j_2 = j_3$, and moreover we impose $j_1 \neq j_2$, which occurs with probability (1 - 1/r); the probability of finding the good values for ε_2 and ε_3 is $1/m^2$. Finally the probability for a point to belong in a 4-cycle with such a pattern is $(1 - 1/r)/(rm)^2$;

Putting together these results, we can bound the probability for a point to be in a useless 4-cycle by $(1-1/r)/(rm)^2 + (1-1/m)^2/mr^3$.

A.4 The Case of t-cycles

For a t-cycle, we rewrite the cancellation of τ as a system of k equations, one for each j_i which actually occurs. We evaluate the probability of $(\tau = 0)$ for every value of k, and then collect the results. Note that k has to be at most t/2, because there has to be at least two terms in each equation, and of course k is at most r. Now let \mathcal{S} be the set of the j_i which actually occur in τ . Then

$$\Pr(\tau = 0) = \sum_{k=1}^{\min(r, t/2)} \Pr(\tau = 0 \mid \#S = k).\Pr(\#S = k).$$

First step: $\Pr(\#S = k) \le \min(\frac{r!k^t}{(r-k)!r^t}, 1)$.

This problem can be expressed as follows: what is the probability to get exactly k distinct elements, when one takes randomly t elements in a set of r elements (we put the element back in the set after each step). This probability is classically (see [8]) $\binom{r}{k} S_2(t,k)/r^t$, where $S_2(t,k)$ is the Stirling number of the second kind. Then we bound $S_2(t,k)$ by $k!k^t$, and obtain the formula.

Second step: $Pr(\tau = 0 \mid \#S = k)$.

Let us make the change of variables

$$\begin{cases} \varepsilon_1^* = \varepsilon_{t-1} \cdots \varepsilon_1 \\ \varepsilon_2^* = \varepsilon_{t-1} \cdots \varepsilon_2 \\ \vdots \\ \varepsilon_{t-1}^* = \varepsilon_{t-1} \end{cases}$$

This transformation is invertible since the ε_i are invertible. The expression for τ becomes $\tau = [\varepsilon_{t-1} \cdots \varepsilon_1] T^{(j_1)} \oplus \cdots \oplus T^{(j_t)} = [\varepsilon_1^*] T^{(j_1)} \oplus \cdots \oplus [\varepsilon_{t-1}^*] T^{(j_{t-1})} \oplus T^{(j_t)} = \bigoplus_{l=1}^k [\mathcal{E}_l] T^{(j_{i_l})}$, where the \mathcal{E}_l are linear expressions in the ε_i^* . When the ε_i^* are randomly chosen among m values, for each l the probability that \mathcal{E}_l is verified is bounded by 1/m. Thus the probability to have a good (t-1)-uple of ε_i is bounded by $1/m^k$, and the result follows.

Note that we used the fact that we have a true t-cycle, because we supposed that the ε_i were randomly chosen, which is not the case if for example we have a 2-cycle (ε_2 has an imposed value in that case).

References

- R. D. Accola. Two theorems on Riemann surfaces with noncyclic automorphism groups. Proc. Amer. Math. Soc., 25:598–602, 1970. 111
- L. M. Adleman, J. DeMarrais, and M.-D. Huang. A subexponential algorithm
 for discrete logarithms over the rational subgroup of the jacobians of large genus
 hyperelliptic curves over finite fields. In L. Adleman and M.-D. Huang, editors,
 ANTS-I, volume 877 of Lecture Notes in Comput. Sci., pages 28–40. SpringerVerlag, 1994. 1st Algorithmic Number Theory Symposium Cornell University,
 May 6-9, 1994. 113, 116
- 3. J. Buhler and N. Koblitz. Lattice basis reduction, Jacobi sums and hyperellitic cryptosystems. *Bull. Austral. Math. Soc.*, 58:147–154, 1998. 113, 114
- D. G. Cantor. Computing in the Jacobian of an hyperelliptic curve. Math. Comp., 48(177):95–101, 1987.
- J. Chao, N. Matsuda, J. Sato, and S. Tsujii. Efficient construction of secure hyperelliptic discrete logarithm problems of large genera. In Proc. Symposium on Cryptography and Information Security, 1997. Fukuoka, Japan. 113, 114
- Y.-M. J. Chen. On the elliptic curve discrete logarithm problem. Preprint, June 1999. 103
- J. H. Cheon, D. H. Lee, and S. G. Hahn. Elliptic curve discrete logarithms and Wieferich primes. Preprint, September 1998. 103
- 8. L. Comtet. Analyse combinatoire. Presses Universitaires de France, 1970. 118
- 9. I. Duursma and K. Sakurai. Efficient algorithms for the jacobian variety of hyperelliptic curves $y^2 = x^p x + 1$ over a finite field of odd characteristic p. In H. Tapia-Recillas, editor, *Proceedings of the "International Conference on Coding Theory, Cryptography and Related Areas"*, volume yyy of *Lecture Notes in Comput. Sci.*, 1999. Guanajuato, Mexico on April, 1998. 111, 113, 114
- P. Flajolet and A. M. Odlyzko. Random mapping statistics. In J.-J. Quisquater, editor, Advances in Cryptology, volume 434 of Lecture Notes in Comput. Sci., pages 329–354. Springer-Verlag, 1990. Proc. Eurocrypt '89, Houthalen, April 10–13. 105
- G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comp., 62(206):865–874, April 1994. 103, 113
- S. D. Galbraith, S. Paulus, and N. P. Smart. Arithmetic on superelliptic curves. Preprint, 1999. 116
- 13. R. Gallant, Lambert, S. Vanstone. Improving R. and paranomalous allelized Pollard lambda search on binary curves. http://www.certicom.com/chal/download/paper.ps, 1998. 104, 107, 109, 110
- 14. P. Gaudry. A variant of the Adleman-DeMarrais-Huang algorithm and its application to small genera. Research Report LIX/RR/99/04, LIX, June 1999. Available at http://www.lix.polytechnique.fr/Labo/Pierrick.Gaudry/. 116
- M. J. Jacobson, N. Koblitz, J. H. Silverman, A. Stein, and E. Teske. Analysis of the Xedni calculus attack. Preprint, February 1999. 103
- E. Kani and M. Rosen. Idempotent relations and factors of Jacobians. Math. Ann., 298:307–327, 1989. 111
- 17. H. J. Kim, J. Cheon, and S. Hahn. Elliptic logarithm over a finite field and the lifting to Q. Preprint, September 1998. 103
- N. Koblitz. A course in number theory and cryptography, volume 114 of Graduate Texts in Mathematics. Springer-Verlag, 1987. 103

- 19. N. Koblitz. Elliptic curve cryptosystems. Math. Comp., 48(177):203–209, January 1987. 103
- N. Koblitz. Hyperelliptic cryptosystems. J. of Cryptology, 1:139–150, 1989. 103, 112, 114, 115
- N. Koblitz. A family of jacobians suitable for discrete log cryptosystems. In S. Goldwasser, editor, Advances in Cryptology - CRYPTO '88, volume 403 of Lecture Notes in Comput. Sci., pages 94-99. Springer-Verlag, 1990. Proceedings of a conference on the theory and application of cryptography held at the University of California, Santa Barbara, August 21-25, 1988. 113, 114
- N. Koblitz. CM-curves with good cryptographic properties. In Joan Feigenbaum, editor, Advances in Cryptology – CRYPTO '91, volume 576 of Lecture Notes in Comput. Sci., pages 279–287. Springer-Verlag, 1992. Santa Barbara, August 12–15.
 110
- H. W. Lenstra, Jr. Factoring integers with elliptic curves. Ann. of Math. (2), 126:649–673, 1987.
- 24. R. Lercier. Finding good random elliptic curves for cryptosystems defined over F_{2^n} . In W. Fumy, editor, *Advances in Cryptology EUROCRYPT '97*, volume 1233 of *Lecture Notes in Comput. Sci.*, pages 379–392. Springer-Verlag, 1997. International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 1997, Proceedings. 113
- 25. R. Lercier and F. Morain. Counting the number of points on elliptic curves over finite fields: strategies and performances. In L. C. Guillou and J.-J. Quisquater, editors, Advances in Cryptology EUROCRYPT '95, volume 921 of Lecture Notes in Comput. Sci., pages 79–94, 1995. International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 1995, Proceedings. 113
- 26. A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curves logarithms to logarithms in a finite field. *IEEE Trans. Inform. Theory*, IT–39(5):1639–1646, September 1993. 103, 112
- 27. A. J. Menezes. *Elliptic curve public key cryptosystems*. Kluwer Academic Publishers, 1993. 103
- V. Miller. Use of elliptic curves in cryptography. In A. M. Odlyzko, editor, Advances in Cryptology – CRYPTO '86, volume 263 of Lecture Notes in Comput. Sci., pages 417–426. Springer-Verlag, 1987. Proceedings, Santa Barbara (USA), August 11–15, 1986. 103
- J. S. Milne. Jacobian varieties. In G. Cornell and J. H. Silverman, editors, Arithmetic Geometry, pages 167–212. Springer-Verlag, 1986. 110
- 30. V. Müller. Fast multiplication on elliptic curves over small fields of characteristic two. J. of Cryptology, 11(4):219–234, 1998. 104, 111
- 31. D. Mumford. Tata lectures on theta II. Birkhauser, 1984. 112
- 32. S. Nakajima. p-ranks and automorphism groups of algebraic curves. Trans. Amer. Math. Soc., 303(2):595–607, October 1987. 111
- 33. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inform. Theory*, IT-24:106–110, 1978. 104
- 34. J. M. Pollard. Monte Carlo methods for index computation (mod p). Math. Comp., 32(143):918-924, July 1978. 104
- 35. J.-J. Quisquater and J.-P. Delescaille. How easy is collision search? application to DES. In J.-J. Quisquater, editor, *Advances in Cryptology*, volume 434 of *Lecture Notes in Comput. Sci.*, pages 429–434. Springer-Verlag, 1990. Proc. Eurocrypt '89, Houthalen, April 10–13. 105

- 36. P. Roquette. Abschätzung der Automorphismenanzahl von Funktionenkörpern bei Primzahlcharakteristik. *Math. Z.*, 117:157–163, 1970. 111
- 37. Y. Sakai and K. Sakurai. Design of hyperelliptic cryptosystems in small characteristic and a software implementation over F₂ⁿ. In K. Ohta and D. Pei, editors, Advances in Cryptology, volume 1514 of Lecture Notes in Comput. Sci., pages 80–94. Springer-Verlag, 1998. Proc. Asiacrypt '98, Beijing, October, 1998. 113, 114
- 38. T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. *Comment. Math. Helv.*, 47(1):81–92, 1998.
- 39. J. Sattler and C. P. Schnorr. Generating random walks in groups. *Ann. Univ. Sci. Budapest. Sect. Comput.*, 6:65–79, 1985. 106, 112
- 40. I. A. Semaev. Evaluation of discrete logarithms in a group of *p*-torsion points of an elliptic curves in characteristic *p. Math. Comp.*, 67(221):353–356, January 1998.
- 41. D. Shanks. Class number, a theory of factorization, and genera. In *Proc. Symp. Pure Math. vol. 20*, pages 415–440. AMS, 1971. 104
- 42. J. H. Silverman. Advanced Topics in the Arithmetic of Elliptic Curves, volume 151 of Grad. Texts in Math. Springer-Verlag, 1994. 111
- 43. J. H. Silverman. The XEDNI calculus and the elliptic curve discrete logarithm problem. Preprint, August 1998. 103
- N. Smart. The discrete logarithm problem on elliptic curves of trace one. Preprint HP-LABS Technical Report (Number HPL-97-128). To appear in *J. Cryptology*, 1997. 103
- N. P. Smart. Elliptic curve cryptosystems over small fields of odd characteristic.
 J. of Cryptology, 12(2):141–151, 1999. 104, 111
- A.-M. Spallek. Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen. PhD thesis, Universität Gesamthochschule Essen, July 1994. 113, 116
- 47. A. Stein and E. Teske. Catching kangaroos in function fields. Preprint, March 1999. 116
- 48. H. Stichtenoth. Uber die Automorphismengruppe eines algebraischen Funktionenkörpers von Primzahlcharakteristik. I. Eine Abschätzung der Ordnung der Automorphismengruppe. Arch. Math. (Basel), 24:527–544, 1973. 111
- E. Teske. Speeding up Pollard's rho method for computing discrete logarithms. In J. P. Buhler, editor, Algorithmic Number Theory, volume 1423 of Lecture Notes in Comput. Sci., pages 541–554. Springer-Verlag, 1998. Third International Symposium, ANTS-III, Portland, Oregon, june 1998, Proceedings. 106
- 50. P. C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *J. of Cryptology*, 12:1–28, 1999. 103, 105
- P. van Wamelen. Examples of genus two CM curves defined over the rationals. Math. Comp., 68(225):307–320, January 1999.
- 52. M. J. Wiener and R. J. Zuccherato. Faster attacks on elliptic curve cryptosystems. In S. Tavares and H. Meijer, editors, Selected Areas in Cryptography '98, volume 1556 of Lecture Notes in Comput. Sci.. Springer-Verlag, 1999. 5th Annual International Workshop, SAC'98, Kingston, Ontario, Canada, August 17-18, 1998, Proceedings. 104, 107, 110

ECC: Do We Need to Count?

Jean-Sébastien Coron^{1,2}, Helena Handschuh^{2,3}, and David Naccache²

École Normale Supérieure
 45 rue d'Ulm, F-75005, Paris, France
 coron@clipper.ens.fr
 Gemplus Card International
 34 rue Guynemer, Issy-les-Moulineaux, F-92447, France
 {coron,handschuh,naccache}@gemplus.com
 École Nationale Supérieure des Télécommunications
 46 rue Barrault, F-75013, Paris, France
 handschu@enst.fr

Abstract. A prohibitive barrier faced by elliptic curve users is the difficulty of computing the curves' cardinalities. Despite recent theoretical breakthroughs, point counting still remains very cumbersome and intensively time consuming.

In this paper we show that point counting can be avoided at the cost of a protocol slow-down. This slow-down factor is quite important (typically $\cong 500$) but proves that the existence of secure elliptic-curve signatures is not necessarily conditioned by point counting.

Keywords: Elliptic curve, point counting, signature.

1 Introduction

Point counting is the most complex part of elliptic-curve cryptography which, despite constant improvements, still remains time-consuming and cumbersome (we refer the reader to [3,4,8,9,13,14,15,11,17,20,21] for a comprehensive bibliography about cardinality counting).

Elliptic-curve cryptosystems that would not require point counting are thus theoretically interesting, although, having taken the decision to design such a scheme, one must overcome three technical difficulties:

- If the number of points on the curve $(\#\mathcal{C})$ is unknown to the participants, the protocol must never involve q, the large prime factor of $\#\mathcal{C}$. This excludes the computation of modular inverses modulo q by the signer and the verifier (recall that DSA signatures involve $s = (m + xr)/k \mod q$ and verifications require $1/s \mod q$).
- Being unknown, $\#\mathcal{C}$ may be accidentally smooth enough to be vulnerable to Pohlig-Hellman attack [18]. An attacker could then undertake the point counting avoided by the designer, factor $\#\mathcal{C}$ and break-down the Discrete Logarithm Problem's complexity into the much easier tasks of solving DLPs in the various subgroups that correspond to the factors of $\#\mathcal{C}$.

• Finally, even if $\#\mathcal{C}$ has a large prime factor q, the choice of the group generator G (e.g. ECDSA's exponentiation base) may still yield a small subgroup vulnerable to discrete logarithm extraction.

Sections 2, 3 and 4 will develop separately each of these issues which will be assembled as a consistent, point-counting-free cryptosystem in section 5. By easing considerably key-generation, our protocol will extend the key-range of elliptic-curve cryptosystems and open new research perspectives.

2 Poupard-Stern's q-free DSA

In Eurocrypt'98, Poupard and Stern [19] presented a DSA-like scheme that combines DLP-based provable security, short identity-based keys, very low transmission overhead and minimal on-line computations. By opposition to other Schnorr-like schemes, Poupard-Stern's protocol uses the order of the multiplicative group q only for system setup (figure 1).

```
System parameters  \begin{array}{c} \text{primes } p \text{ and } q \text{ such that } q | (p-1) \\ g \in \mathbb{Z}/p\mathbb{Z} \text{ of order } q \\ \text{a hash function } h : \{0,1\}^* \to \mathbb{Z}/q\mathbb{Z} \\ \text{Key generation} \\ & \text{secret } : x \in_R \mathbb{Z}/q\mathbb{Z} \\ \text{public } : y = g^{-x} \text{ mod } p \\ \\ & \text{Signature} \\ \\ & \text{Signature} \\ & \text{pick a large random } k \\ & r = g^k \text{ mod } p \\ & s = k + x \times h(m,r) \\ & \text{signature } : \{r,s\} \\ \\ \text{Verification} \\ & \text{check that } r \overset{?}{=} g^s y^{h(m,r)} \text{ mod } p \\ \\ \end{array}
```

Fig. 1. Poupard-Stern signatures.

We refer the reader to [19] for a precise definition of the system parameters (e.g. the size of k), a formal security proof and a description of the scheme's implementation trade-offs.

Elliptic-curve generalization is straightforward: let p be the size of the underlying field (or ring) on which the curve is defined (a prime, an RSA modulus or 2^n); when p is a prime or an RSA modulus the equation of the curve C, characterized by a and b, is given by $y^2 = x^3 + ax + b$; the curve will be defined by

 $y^2 + xy = x^3 + ax + b$ when the underlying field is $GF(2^n)$. In the elliptic curve Pourpard-Stern signature scheme, p-1 and q are respectively replaced by $\#\mathcal{C}$ and one of its large prime factors (figure 2).

```
System parameters
                         a prime q
                         an elliptic curve \mathcal{C} such that q \mid \#\mathcal{C}
                         G \in \mathcal{C} of order q
                         a hash function h: \{0,1\}^* \to \mathbb{Z}/q\mathbb{Z}
Key generation
                         secret: x \in_R \mathbb{Z}/q\mathbb{Z}
                         public: Y = -x\bar{G}
Signature
                         pick a large random k
                         R = kG = (x_R, y_R)
                         s = k + x \times h(m, x_R)
                         signature : \{x_R, s\}
Verification
                         compute R' = sG + h(m, x_R)Y = (x_{R'}, y_{R'})
                         check that x_R \stackrel{?}{=} x_{R'}
```

Fig. 2. Elliptic-curve Poupard-Stern signatures.

Poupard and Stern's security proof can be extended, *mutatis mutandis*, to the elliptic-curve variant; the proof can be consulted in the appendix.

We will now suppress from the above protocol the last references to q; care should be taken to underline that we do not claim yet that the resulting protocol (figure 3) is secure.

3 The Expected Smoothness of $\#\mathcal{C}$

As an inescapable consequence of our modification, $\#\mathcal{C}$ may now be smooth enough to be at Pohlig-Hellman's reach. An attacker could then perform the point counting, factor $\#\mathcal{C}$ and reduce the DLP's complexity into the much easier tasks of solving DLPs in the various subgroups that correspond to the different factors of $\#\mathcal{C}$. Moreover, even if $\#\mathcal{C}$ has a large prime factor it may still be divisible by a product π of small primes, allowing the adversary to find a portion of the secret key $(x \mod \pi)$ using Pohlig-Hellman. Using Hesse's theorem, we set $L = \log_2 \lfloor p + 1 - 2\sqrt{p} \rfloor$ and deliberately accept that only ℓ bits of the L-bit secret key will actually remain unknown to the attacker.

System parameters $\begin{array}{c} \text{a random elliptic curve } \mathcal{C} \\ G \in_R \mathcal{C} \\ \text{a hash function } h: \{0,1\}^* \to \{0,1\}^L \\ \text{Key generation} \\ \text{secret}: x \in_R \{0,1\}^L \\ \text{public}: Y = -xG \\ \end{array}$ Processing $\begin{array}{c} \text{pick a large random } k \\ R = kG = (x_R, y_R) \\ s = k + x \times h(m, x_R) \\ \text{output}: \{x_R, s\} \\ \end{array}$ Verification $\begin{array}{c} \text{compute } R' = sG + h(m, x_R)Y = (x_{R'}, y_{R'}) \\ \text{check that } x_R \stackrel{?}{=} x_{R'} \end{array}$

Fig. 3. q-free EC variant of Poupard-Stern's protocol.

We consider that a curve is weak if all the factors of $\#\mathcal{C}$ are smaller than 2^{ℓ} (i.e. $\#\mathcal{C}$ is 2^{ℓ} -smooth) where ℓ is a security parameter. The odds of such an event are analyzed in this section under the assumption that $\#\mathcal{C}$ is uniformly distributed over $[p+1-2\sqrt{p},p+1+2\sqrt{p}]$.

Defining $\psi(x,y) = \#\{n < x, \text{ such that } n \text{ is } y\text{-smooth}\}$, it is known [5,6,7] that, for large x, the ratio :

$$\frac{\psi(x,\sqrt[t]{x})}{x}$$

is equivalent to Dickman's function defined by:

$$\rho(t) = \begin{cases} 1 & \text{if } 0 \le t \le 1 \\ \rho(n) - \int_n^t \frac{\rho(v-1)}{v} dv & \text{if } n \le t \le n+1 \end{cases}$$

 $\rho(t)$ is thus an approximation of the probability that a $\ell \times t$ -bit number is 2^{ℓ} -smooth; table 1 summarizes the value of ρ for $2 \le t \le 10$.

Since $\rho(t)$ is not easy to compute, we will use throughout this paper the exact formula for $t \le 10$ and de Bruijn's asymptotic approximation [1,2] for t > 10:

$$\rho(t) \cong (2\pi t)^{-1/2} \exp\left(\gamma - t\zeta + \int_0^{\zeta} \frac{e^s - 1}{s} ds\right)$$

where ζ is the positive solution of $e^{\zeta} - 1 = t\zeta$ and γ is Euler's constant.

ſ	t	2	3	4	5	6	7	8	9	10
	$\rho(t)$	3.07e-1	4.86e-2	4.91e-3	3.54e-4	1.96e-5	8.75e-7	3.23e-8	1.02e-9	2.79e-11

Table 1.
$$\rho(t)$$
 for $2 \le t \le 10$.

Table 1 shows that the proportion of weak curves is too high for immediate use: values of t, such as 2 and 3, which would respectively yield 320 and 480-bit field size for $\ell = 160$, correspond to a percentage of 0.3 and 0.05 weak curves. In section 5, we will propose a signature strategy that decreases exponentially these probabilities.

As pointed out earlier, the above assumes that $\#\mathcal{C}$ is distributed uniformly over $[p+1-2\sqrt{p},p+1+2\sqrt{p}]$. A more accurate result, valid for prime p, was proved by Lenstra in [12]:

Theorem 1. Denoting by #' the number of isomorphism classes of elliptic curves, and $\Delta(S) = \#'\{\text{elliptic curves } \mathcal{E} \text{ over } F_p \text{ such that } \#\mathcal{E} \in S \subset \mathbb{N}\}$ there exist effectively computable positive constants c_1 , c_2 such that for each prime p > 3, the following holds:

- if for all $s \in S$, $|s (p+1)| \le 2\sqrt{p}$ then $\Delta(S) \le c_1 \# S\sqrt{p}(\log p)(\log \log p)^2$
- if for all $s \in S$, $|s (p+1)| \le \sqrt{p}$ then $\Delta(S) \ge c_2 \sqrt{p} (\#S 2) / \log p$

Since all classes have a number of representatives which is roughly p, Lenstra's theorem basically claims that by taking a curve at random, the probability τ_S that its cardinality lies in S satisfies the inequality:

$$\frac{c_3}{\log p} \le \frac{\tau_S}{\pi_S} \le c_4 (\log p) (\log \log p)^2$$

where π_S denotes the probability of picking an element of S at random in the interval $[p-\sqrt{p},p+\sqrt{p}]$. The theorem indicates that (at least if p is prime) when C is random, the proportion of weak-curves respects Dickman's estimate. We consider this as heuristically satisfactory for further build-up.

4 The Expected Order of the Generator G

Even when $\#\mathcal{C}$ has an prime factor larger than ℓ bits, G could still yield a small subgroup, which would again weaken the scheme.

We refer the reader to [16] for the following theorem :

Theorem 2. The set of points of an elliptic curve is an abelian group which is either a cyclic group or the product of two cyclic groups.

Let q be a large prime factor of $r = \#\mathcal{C}$ of multiplicity 1.

- Assume that C is a cyclic abelian group, isomorphic to $\mathbb{Z}/r\mathbb{Z}$, with generator $g \in C$. The order d of a random $G = g^{\alpha}$ is given by $d = r/\gcd(r, \alpha)$. Therefore q does not divide d if and only if α is a multiple of q. The probability that the order of a random G is not divisible by q is thus 1/q.
- Assume that \mathcal{C} is the product of two cyclic abelian groups, then \mathcal{C} is isomorphic to some product $\mathbb{Z}/r_1\mathbb{Z} \times \mathbb{Z}/r_2\mathbb{Z}$ where r_2 divides r_1 and $r_1r_2 = r$. For a large prime factor q of r (with multiplicity 1), q divides r_1 but not r_2 . Therefore q divides the order of an element of the curve if and only if q divides the order of this element with respect to $\mathbb{Z}/r_1\mathbb{Z}$. This leads back to the first case, and the probability that the order of a random G is not divisible by q is 1/q again.

In both cases, the probability that a random choice for G yields a small subgroup is negligible.

5 The New Scheme

The new protocol iterates the signature on a few curves in order to reduce (below an $\epsilon = 2^{-\ell/2}$) the probability that all curves will be smooth (figure 4):

```
System parameters
                           \sigma random elliptic curves \mathcal{C}_1, \ldots, \mathcal{C}_{\sigma}
                           \sigma random points G_1, \ldots, G_{\sigma} such that G_i \in \mathcal{C}_i
                           a hash function h: \{0,1\}^* \to \{0,1\}^L
Key generation
                           secret : \sigma random integers x_i \in_R \{0,1\}^L
                           public: \sigma points Y_i = -x_i G_i such that Y_i \in \mathcal{C}_i
Signature
                           for i = 1 to \sigma
                                 pick a large random k_i
                                 compute R_i = k_i G_i \in \mathcal{C}_i = (x_{R_i}, y_{R_i})
                                 compute s_i = k_i + x_i \times h(m, x_{R_i})
                           signature : \{\{x_{R_1}, s_1\}, \dots, \{x_{R_{\sigma}}, s_{\sigma}\}\}
Verification
                           for i = 1 to \sigma
                                  Compute R'_{i} = s_{i}G_{i} + h(m, x_{R_{i}})Y_{i} = (x_{R'_{i}}, y_{R'_{i}})
                                  Check that x_{R'_i} \stackrel{?}{=} x_{R_i}
```

Fig. 4. q-free elliptic-curve Poupard-Stern signatures.

The number of necessary iterations σ is given by :

$$\rho(|p|/\ell)^{\sigma} \le \epsilon \quad \Rightarrow \quad \sigma = \left\lceil \frac{\ell}{2\log \rho(|p|/\ell)} \right\rceil$$

and is summarized in table 2 for $\ell = 160$.

The slow-down factor γ between the elliptic curve Poupard-Stern signature scheme and the new scheme (signature generation times) is due to the iteration of the signature on σ curves and the increased complexity of point operations over bigger underlying fields. Since the time complexity of elliptic curve scalar multiplication is in $\mathcal{O}(|p|^3)$, γ is basically given by:

$$\gamma = \sigma \times \left(\frac{|p|}{\ell}\right)^3$$

The slow-down factor is summarized in table 2 for $\ell = 160$.

# of iterations σ	size of p	slow-down	# of iterations σ	size of p	slow-down
20	460 bits	474	10	654 bits	683
19	471 bits	486	9	693 bits	732
18	484 bits	499	8	740 bits	791
17	497 bits	509	7	798 bits	868
16	513 bits	526	6	873 bits	977
15	529 bits	542	5	973 bits	1125
14	548 bits	562	4	1115 bits	1352
13	570 bits	588	3	1337 bits	1746
12	594 bits	613	2	1757 bits	2646
11	622 bits	645	1	2800 bits	5355

Table 2. Protocol trade-offs for $\ell = 160$.

Letting alone the factor γ , the verification times of the new scheme are also slower than usual ECC ones (e.g. ECDSA) because of the additional increase in the size of s due to the Poupard-Stern construction.

Note that Poupard-Stern's security proof will still apply to (at least one of) our curves with probability greater than $1-\epsilon \cong 1$. Surprisingly, instances will be either provably secure against existential forgery under adaptive chosen message attacks (probability greater than $1-\epsilon$) or insecure (probability lower than $\epsilon = 2^{-\ell/2}$) without transiting through intermediate gray areas where security is only conjectured (our ϵ is, of course, not related to [19]'s one).

Although the security proof has not been extended to the case where all curves have the same system parameters (identical p, intersection in G), we conjecture that the resulting scheme (figures 5 and 6) is still secure.

Secret parameters $(x_i \text{ and } k_i)$ must however remain distinct for every curve, given the (deliberately accepted) risk that the DLP might be easy on *some* of our curves.

```
System parameters
                          \sigma elliptic curves \mathcal{C}_1, \ldots, \mathcal{C}_{\sigma} intersecting in G
                          a hash function h: \{0,1\}^* \to \{0,1\}^L
Key generation
                         secret : \sigma random integers x_i \in_R \{0,1\}^L
                         public: \sigma points Y_i = -x_i G such that Y_i \in \mathcal{C}_i
Signature
                          for i = 1 to \sigma
                                pick a large random k_i
                                compute R_i = k_i G \in \mathcal{C}_i = (x_{R_i}, y_{R_i})
                                compute s_i = k_i + x_i \times h(m, x_{R_i})
                          signature : \{\{x_{R_1}, s_1\}, \dots, \{x_{R_{\sigma}}, s_{\sigma}\}\}
Verification
                          for i = 1 to \sigma
                                Compute R'_i = s_i G + h(m, R_i) Y_i = (x_{R'_i}, y_{R'_i})
                                Check that x_{R'_i} \stackrel{?}{=} x_{R_i}
```

Fig. 5. q-free elliptic-curve Poupard-Stern signatures (common G).

It is important to point-out that, due to our probabilistic design, the signer must generate C_1, \ldots, C_{σ} or (at least) make sure that the authority can exhibit a random seed (similar to the DSA's certificate of proper key generation) that yields all the curves' parameters by hashing.

6 Extensions and Variants

The scheme can be improved in many ways: by hashing $x_i = h'(x,i)$ and $k_i = h''(k,i)$ one can make the economy of $\sigma - 1$ secret keys and session randoms; a particularly efficient variant consists in grouping $\{R_1, \ldots, R_{\sigma}\}$ in a single digest (figure 7); the scheme can, of course, be implemented on any group.

Note that when p is an RSA modulus (hereafter n), life becomes much harder for the attacker who must (in our present state of knowledge) factor n (equivalent to point counting [10]), compute the orders d_1 and d_2 of the curve modulo the prime factors of n, factor d_1 and d_2 and compute the exact order of G as a multiplicative combination of the prime factors of d_1 and d_2 .

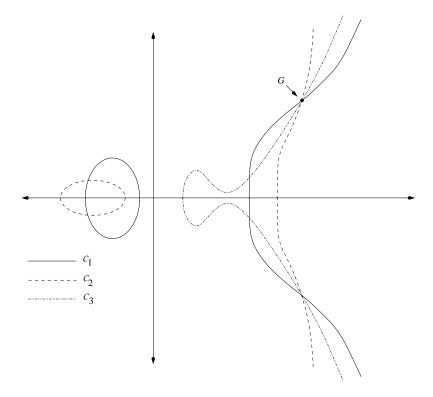


Fig. 6. System configuration (intersecting curves) for $\sigma = 3$.

The overwhelming security contribution comes from the factorisation of n although when this calculation comes to an end, the attacker may face a (non-smooth) curve where the DLP is hard. The attacker's success chances are consequently reduced to:

$$\epsilon' = \rho \left(\frac{|n|}{2\ell}\right)^2$$

for one curve and

$$\epsilon'' = \epsilon'^{\sigma} = \rho \left(\frac{|n|}{2\ell}\right)^{2\sigma}$$

for the σ curves. This indicates an interesting way of squeezing more complexity out of RSA moduli : since (in our present state of knowledge) smooth curves can not be spotted without factoring n, the inverse of ϵ'' represents a strengthening factor that multiplies the attacker's effort by a factor depending on |n| and σ (table 3 for $\ell=160$).

¹ under the discrete logarithm assumption.

```
System parameters
                             \sigma elliptic curves \mathcal{C}_1, \ldots, \mathcal{C}_{\sigma} intersecting in G
                             a hash function h: \{0,1\}^* \to \{0,1\}^L
Key generation
                            secret : \sigma random integers x_i \in_R \{0,1\}^L
                            public: \sigma points Y_i = -x_i G such that Y_i \in \mathcal{C}_i
Signature
                             for i = 1 to \sigma
                                   pick a large random k_i
                                   compute R_i = k_i G \in \mathcal{C}_i = (x_{R_i}, y_{R_i})
                            r = h(m, x_{R_1}, \dots, x_{R_{\sigma}})
                             for i = 1 to \sigma
                                   compute s_i = k_i + x_i \times r
                            signature : \{r, s_1, \ldots, s_{\sigma}\}
                             for i = 1 to \sigma
                            compute R_i' = s_i G + r Y_i = (x_{R_i'}, y_{R_i'})
check that r \stackrel{?}{=} h(m, x_{R_1'}, \dots, x_{R_\sigma'})
```

Fig. 7. q-free elliptic-curve Poupard-Stern signatures (common G and r).

$-\log_2 \text{ factor } \setminus$	n = 512	n = 768	n = 1024
$\sigma = 1$	1.8	5.3	9.9
$\sigma = 2$	3.6	10.7	19.9
$\sigma = 3$	5.5	16.0	29.8
$\sigma = 4$	7.3	21.4	39.8
$\sigma = 5$	9.1	26.8	49.8
$\sigma = 6$	11.0	32.1	59.7
$\sigma = 7$	12.8	37.5	69.7

Table 3. Strengthening factors for $\ell = 160$ and $1 \le \sigma \le 7$.

7 Acknowledgements

The authors are grateful to Jacques Stern for motivating and following the evolution this work; we also thank him for his insights into several mathematical details and for kindly providing the extended proof given in the appendix.

References

- 1. N. de Bruijn, On the number of positive integers $\leq x$ and free of prime factos $\geq y$, Indagationes Mathematicae, vol. 13, pp. 50–60, 1951. 125
- 2. N. de Bruijn, On the number of positive integers $\leq x$ and free of prime factos $\geq y$, II, Indagationes Mathematicae, vol. 28, pp. 236–247, 1966. 125
- 3. J.-M. Couveignes, L. Dewaghe & F. Morain, Isogeny cycles and the Schoof-Elkies-Atkin algorithm, Rapport de recherche LIX/RR/96/03, Laboratoire d'informatique de l'École Polytechnique, 1996. 122
- J.-M. Couveignes & F. Morain, Schoof's algorithm and isogeny cycles, LNCS 877, ANTS-I, Proceedings of first algorithmic number theory symposium, Springer-Velrag, pp. 43–58, 1994. 122
- K. Dickman, On the frequency of numbers containing prime factors of a certain relative magnitude, Arkiv för matematik, astronomi och fysik, vol. 22A(10), pp. 1–14, 1930. 125
- J. Dixon, Asymptotically fast factorization of integers, Mathematics of computation, vol. 36(153), pp. 255–260, 1981. 125
- H. Halberstam, On integers whose prime factors are small, Proceedings of the London mathematical society, vol. 3(21), pp. 102–107, 1970. 125
- 8. E. Howe, On the group orders of elliptic curves over finite fields, Compositio mathematica, vol. 85, pp. 229–247, 1993. 122
- 9. N. Koblitz, Primality of the number of points on an elliptic curve over a finite field, Pacific journal of mathematics, vol. 131, pp. 157–165, 1988. 122
- 10. N. Kunihiro & K. Koyama, Equivalence of counting the number of points on elliptic curve over the ring \mathbb{Z}_n and factoring n, LNCS 1403, Advances in cryptology proceedings of Eurocrypt'98, Springer-Verlag, pp. 47–58, 1998. 129
- G. Lay & H. Zimmer, Constructing elliptic curves with given group order over large finite fields, LNCS 877, ANTS-I, Proceedings of first algorithmic number theory symposium, Springer-Velrag, pp. 250–263, 1994.
- 12. H. Lenstra Jr., Factoring integers with elliptic curves, Ann. math., vol. 126, pp. 649–673, 1987. 126
- R. Lercier, Computing isogenies in GF(2ⁿ), LNCS 1122, ANTS-II, Proceedings of 2-nd algorithmic number theory symposium, Springer-Velrag, pp. 197–212, 1996.
 122
- R. Lercier & F. Morain, Counting the number of points on elliptic curves over finite fields: strategies and performances, LNCS 921, Advances in cryptology proceedings of Eurocrypt'95, Springer-Verlag, pp. 79–94, 1995.
- 15. R. Lercier & F. Morain, Counting the number of points on elliptic curves over F_{p^n} using Couveigne's algorithm, Rapport de recherche LIX/RR/95/09, Laboratoire d'informatique de l'École Polytechnique, 1995. 122
- A. Menezes, Elliptic curve public key cryptosystems, Kluwer academic publishers, pp. 25, 1983. 126
- 17. A. Menezes, S. Vanstone & R. Zuccharato, Counting points on elliptic curves over F_{2m} , Mathematics of computation, vol. 60(201), pp. 407–420, 1993. 122

- S. Pohlig & M. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, IEEE Transactions on Information Theory, Vol. 24, pp. 106-110, 1978. 122
- G. Poupard & J. Stern, A practical and provably secure design for on the fly authentication and signature generation, LNCS 1403, Advances in cryptology proceedings of Eurocrypt'98, Springer-Verlag, pp. 422–436, 1998. 123, 128, 133
- R. Schoof, Elliptic curves over finite fields and the computation of square roots mod p, Mathematics of computation, vol. 44, pp. 483–494, 1985.
- R. Schoof, Counting points on elliptic curves over finite fields, CACM, vol. 21(2), pp. 120-126, 1978.

APPENDIX

Using Poupard and Stern's notations, the following is a generalization of [19]'s security proof :

Theorem 3. Assume that $kS\tau/X$ and 1/k are negligible. If an existential forgery of the signature scheme under adaptive chosen message attack has a nonnegligible success probability then the discrete logarithm on elliptic curves can be computed in a time polynomial in |q|.

The proof is based on the same 3-fork variant of Pointcheval-Stern's forking lemma. However, there is a technical difficulty; in the modular case studied by Poupard and Stern, the authors deal with an RSA modulus n = pq, assuming that g is of order $\lambda(n) = \text{GCD}(p-1, q-1)$. Their proof includes three steps:

- 1. compute a multiple L of $\lambda(n)$.
- 2. factor n, using L and a number-theoretic algorithm due to Miller.
- 3. finally, use the 3-fork variant of the forking lemma to yield a couple of relations involving the unknown key s:

$$\alpha s + \beta = 0 \mod \lambda(n)$$
 and $\alpha' s + \beta' = 0 \mod \lambda(n)$

such that for some polynomial B, which only depends on the machine which presumably performs the existential forgery, $GCD(\alpha, \alpha') \leq B$; from these equations, s can be computed in polynomial time.

In the elliptic curve case, there is no analog to step 2; which requires a further twist :

- 1. compute a multiple ρ of the (unknown) order r of G, which is approximately |X| + |k|-bit long.
- 2. use the forking lemma's 3-fork variant to yield a couple of relations involving s:

$$\alpha s + \beta = 0 \mod \lambda(n)$$
 and $\alpha' s + \beta' = 0 \mod \lambda(n)$

such that for some polynomial B, which only depends on the machine which presumably performs the existential forgery, $\mathrm{GCD}(\alpha,\alpha') \leq B$. Furthermore, we cancel all primes smaller than B from ρ . From these relations and ρ , one can compute a substitute to s which satisfies V=-sG without being in the proper range.

3. Finally, we show that an algorithm which computes a substitute of s and a multiple of r with significant probability can be turned into an algorithm which computes the proper value of s:

Lemma 1. An algorithm \mathcal{A} which computes with significant probability a fixed-size multiple ρ of the unknown order r of G and a substitute to the secret key $s < \rho$ can be turned into an algorithm \mathcal{B} which computes the proper value of s.

PROOF Let ϵ be the success probability of \mathcal{A} and fix $\delta = \epsilon/|\rho|$. By induction on $|\rho|$, we show how to design an algorithm \mathcal{B} which discloses the actual key with probability at least δ : Apply \mathcal{A} to V = -sG, where s is in the proper range for keys. \mathcal{A} could either output s with probability δ (in which case the proof is complete) or it outputs a substitute $s' \neq s$ with probability bigger than $(\rho - 1)\delta$. In this case, we can consider s' - s and $\rho - s' + s$; both are multiples of r and one of them (hereafter ρ') is smaller than $\rho/2$. Note that \mathcal{A} produces, with probability $\delta|\rho'|$ a multiple ρ' of r. Furthermore, it also produces substitute keys smaller than ρ' , since one can always replace a substitute s by s mod ρ' ; we can thus apply the inductive hypothesis, which completes the proof.

Elliptic Scalar Multiplication Using Point Halving

Erik Woodward Knudsen

De La Rue Card Systems erik.knudsen@fr.delarue.com

Abstract. We describe a new method for conducting scalar multiplication on a non-supersingular elliptic curve in characteristic two. The idea is to replace all point doublings in the double-and-add algorithm with a faster operation called point halving.

1 Introduction

The security of cryptosystems like the Diffie-Hellman scheme is based on the intractability of the Discrete Logarithm Problem of the underlying group. For most elliptic curves defined over finite fields the Discrete Logarithm Problem is believed to be hard to solve and for this reason they are interesting in cryptography. The most time consuming part of the Diffie-Hellman key exchange protocol is multiplication of a point on the curve not known in advance by a random scalar. We will only discuss curves defined over fields of characteristic two; a popular choice for implementations since addition in such a field corresponds to the exclusive-or operation. It is known that scalar multiplication can be speeded up on a curve which is defined over a field of small cardinality ([Koblitz], [Meistaff], [Muller1]) using the Frobenius morphism. The curves can be chosen such that no known attack applies to them. However, at least principally, it is of course preferable to be able to choose the curve which one wants to use from as general a class of curves as possible. The method described in this paper applies in its fastest version to half of the elliptic curves. Moreover, from a cryptographic point of view it is the "better" half. Before giving the principle of the method we formulate the basic concepts. See for example [Silverman] for an introduction to the theory of elliptic curves.

Let n be a fixed integer. Let \mathbf{F}_{2^n} denote the field with 2^n elements and let $\overline{\mathbf{F}}_{2^n}$ denote the algebraic closure of \mathbf{F}_{2^n} . Let \mathcal{O} denote the point at infinity. By a non-supersingular elliptic curve E defined over \mathbf{F}_{2^n} we mean the set

$$E = \{(x,y) \in \overline{\mathbf{F}}_{2^n} \times \overline{\mathbf{F}}_{2^n} \mid y^2 + xy = x^3 + ax^2 + b\} \cup \{\mathcal{O}\} \quad a,b \in \mathbf{F}_{2^n}, \ b \neq 0$$

It is well known that E can be equipped with an abelian group structure where the point at infinity is the neutral element. It is customary to call the elements

of E for points. We will work with the finite subgroup of E

$$E(\mathbf{F}_{2^n}) = \{(x, y) \in \mathbf{F}_{2^n} \times \mathbf{F}_{2^n} \mid y^2 + xy = x^3 + ax^2 + b\} \cup \{\mathcal{O}\}$$

That is the \mathbf{F}_{2^n} -rational points in E. For any $m \in \mathbf{N}$ we can define the multiplication-by-m map

$$[m]: E \to E$$

$$P + \underbrace{-P + \dots + P}_{\text{(m times)}}$$

and for m=0:

$$\forall P \in E : [0]P = \mathcal{O}.$$

The kernel of the multiplication-by-m map is denoted by E[m]. The points of the group E[m] are also called the m-torsion points of E. The group structure of the m-torsion points is well known. We will only be interested in the case where m is a power of two. In this case we have:

$$\forall k \in \mathbf{N} : E[2^k] \simeq \mathbf{Z}/2^k \mathbf{Z}$$

We will use the notation T_{2^k} for a point of order 2^k . Since T_2 is contained in $E(\mathbf{F}_{2^n})$ and since $E(\mathbf{F}_{2^n})$ is a finite subgroup of E it has the structure:

$$E(\mathbf{F}_{2^n}) = G \times E[2^k]$$

where G is a group of odd order and $k \geq 1$. When k = 1 we will say that the curve has minimal two-torsion.

After these preliminaries we are ready to explain the aim of the paper. The multiplication-by-two map, denoted by [2], which we will also call the doubling map, is not an injective function when defined on E or $E(\mathbf{F}_{2^n})$ because it has kernel $E[2] = \{\mathcal{O}, T_2\}$. On the other hand, if we restrict the domain of the doubling map to a subgroup $G \subset E(\mathbf{F}_{2^n})$ of odd order the map is a bijection. Consequently, on this subgroup the doubling map has an inverse map which we call the halving map:

We will write $[\frac{1}{2}]P$ for the point in G which the doubling map sends to P. For all $i \geq 1$ we will write

$$\left[\frac{1}{2^{i}}\right] := \left[\frac{1}{2}\right] \circ \dots \circ \left[\frac{1}{2}\right]$$

for the *i*-fold composition of the halving map. The halving map is interesting in connection with elliptic scalar multiplication for the following reason: it is possible to replace all point doublings used when performing a scalar multiplication

by point halvings. As we shall see the halving map is considerably faster to evaluate than the doubling map on a curve with minimal two-torsion when working in affine coordinates. From a cryptographic viewpoint it is good to have as many curves to choose from as possible and it is customary to use a curve for which the two-torsion of $E(\mathbf{F}_{2^n})$ is either minimal or isomorphic to $\mathbf{Z}/4\mathbf{Z}$. We shall see in Appendix A that for a given field \mathbf{F}_{2^n} the curves with minimal two-torsion constitutes exactly half of all the curves defined over \mathbf{F}_{2^n} . Therefore, although not completely general, the method described applies in its fastest version to a large class of the curves which are interesting in connection with cryptography. The method can always be implemented if the field elements are represented in a normal basis. If a polynomial basis is used the storage requirements are in the order of magnitude $O(n^2)$ bits.

In section 2 we show how to compute $\left[\frac{1}{2}\right]P \in G$ from $P \in G$. In section 3 suggestions are given for fast computations. In section 4 we show how to replace doublings by halvings when performing a scalar multiplication and in section 5 we discuss the expected improvements in running time due to this replacement.

2 Point Halving

Representations of Points

We will use two representations: The usual affine representation of a point

$$P = (x, y)$$

and the representation

$$(x, \lambda_P)$$
 where $\lambda_P = x + \frac{y}{x}$.

From the second representation we can evaluate $y = x(x + \lambda_P)$ using one multiplication. The idea is then, when performing a scalar multiplication, to save field multiplications by performing intermediate results using the representation (x, λ_P) and only determining the second coordinate of the affine representation in the very end.

Point Halving

Given a point P in G we want to calculate $\left[\frac{1}{2}\right]P$. To do this let $P=(x,y)=(x,x(x+\lambda_P))\in G$ and $Q=(u,v)=(u,u(u+\lambda_Q))\in E(\mathbf{F}_{2^n})$ denote points such that [2]Q=P. The doubling formulas are given by ([IEEE]):

$$\lambda_Q = u + \frac{v}{u} \tag{1}$$

$$x = \lambda_Q^2 + \lambda_Q + a \tag{2}$$

$$y = (x+u)\lambda_Q + x + v \tag{3}$$

Multiplying (1) by u and substituting the value of v from (1) into (3) this is rewritten to:

$$v = u(u + \lambda_Q)$$

$$\lambda_Q^2 + \lambda_Q = a + x$$

$$y = (x + u)\lambda_Q + x + u^2 + u\lambda_Q = u^2 + x(\lambda_Q + 1)$$

Remembering that $y = x(x + \lambda_P)$ we get:

$$\lambda_Q^2 + \lambda_Q = a + x$$
 (i)

$$u^2 = x(\lambda_Q + 1) + y = x(\lambda_Q + \lambda_P + x + 1)$$
 (ii)

$$v = u(u + \lambda_Q)$$
 (iii)

With input $P = (x, y) = (x, x(x + \lambda_P))$ in either affine coordinates or the representation (x, λ_P) this system of equations determines the two points

$$\left[\frac{1}{2}\right]P \in G$$
 and $\left[\frac{1}{2}\right]P + T_2 \in E(\mathbf{F}_{2^n})\backslash G$

which are mapped to P by the doubling map. We want to be able to distinguish between them. We start by considering curves with minimal two-torsion:

Theorem 1. Let E be a curve with minimal two-torsion. Let $P \in E(\mathbf{F}_{2^n}) = G \times \{\mathcal{O}, T_2\}$ be an element of odd order. Let Q be a point such that

$$Q \in \{ [\frac{1}{2}]P, [\frac{1}{2}]P + T_2 \}$$

and let Q_1 denote either one of the two points in E for which $[2]Q_1 = Q$. We then have the necessary and sufficient condition

$$Q = \left[\frac{1}{2}\right]P \Leftrightarrow Q_1 \in E(\mathbf{F}_{2^n})$$

Proof. Q_1 is determined by applying the formulas (i),(ii) and (iii) to Q. Q equals either $\left[\frac{1}{2}\right]P$ or $\left[\frac{1}{2}\right]P+T_2$. By applying the formulas (i),(ii) and (iii) to $\left[\frac{1}{2}\right]P$ we get the two points

$$[\frac{1}{4}]P, [\frac{1}{4}]P + T_2 \in E(\mathbf{F}_{2^n})$$

which are in $E(\mathbf{F}_{2^n})$. Let T_4 and $[3]T_4$ denote the two points of order four in E. Applying the formulas (i),(ii) and (iii) to $[\frac{1}{2}]P + T_2$ we get

$$\left[\frac{1}{4}\right]P + T_4, \left[\frac{1}{4}\right]P + \left[3\right]T_4 \not\in E(\mathbf{F}_{2^n})$$

The points are not in $E(\mathbf{F}_{2^n})$ because $T_4 \not\in E(\mathbf{F}_{2^n}) = G \times \{\mathcal{O}, T_2\}$

Theorem 1 tells us that we on a curve with minimal two-torsion can check whether $Q = [\frac{1}{2}]P$ or $Q = [\frac{1}{2}]P + T_2$ by checking whether the coordinates of Q_1 are in \mathbf{F}_{2^n} or in a field extension. Since Q_1 is determined by the equations

(i), (ii) and (iii) we examine these for operations which are not internal to the field. Solving the second degree equation in (i) is one such operation, and it is in fact the only one: it is true that we also have to calculate a square root to calculate the first coordinate of Q_1 , but in characteristic two taking a square root is an operation internal to the field. We thus have:

$$Q = (u, v) = \left[\frac{1}{2}\right]P \Leftrightarrow \exists \lambda \in \mathbf{F}_{2^n} : \lambda^2 + \lambda = a + u$$

Since taking a square root is an operation internal to the field we can state this necessary and sufficient condition in another way:

$$Q = (u, v) = \left[\frac{1}{2}\right]P \Leftrightarrow \exists \lambda \in \mathbf{F}_{2^n} : \lambda^2 + \lambda = a^2 + u^2$$

Using this last condition will optimize the algorithm given below if the time to compute a square root is non-negligible.

Given $P \in G$, the two solutions to (i) are $\lambda_{\left[\frac{1}{2}\right]P}$ and $\lambda_{\left[\frac{1}{2}\right]P} + 1$ and we see from (ii) that the first coordinates of the corresponding candidates are u and $u + \sqrt{x}$. We have justified that we can calculate $\left[\frac{1}{2}\right]P$ in the following manner on a curve with minimal two-torsion:

Point Halving Algorithm

 $\frac{\text{Input: }P=(x,y)=(x,x(x+\lambda_P))\in G \text{ represented either as }(x,y) \text{ or as }(x,\lambda_P)}{\text{Output: } [\frac{1}{2}]P=(u,v)\in G \text{ represented as }(u,\lambda_{[\frac{1}{2}]P}) \\ \text{Method: }$

- 1. Compute a solution $\lambda_{\lceil \frac{1}{2} \rceil P}$ from (i).
- 2. Compute the corresponding u^2 from (ii).
- 3. Check if there exists $\lambda \in \mathbf{F}_{2^n}$ such that $\lambda^2 + \lambda = a^2 + u^2$.
- 4. If such a λ does not exist then compute $u^2:=u^2+x$ and $\lambda_{[\frac{1}{2}]P}:=\lambda_{[\frac{1}{2}]P}+1$
- 5. Calculate $u := \sqrt{u^2}$.
- 6. Output $(u, \lambda_{\left[\frac{1}{2}\right]P})$.

If a^2 is precomputed and stored the algorithm requires solving 1 second degree equation (in 1)

1 multiplication (in 2)

1 check (in 3)

1 square root (in 5)

and if v is to be evaluated using (iii) one extra multiplication is required. We see that if we have to perform k consecutive halvings we can save k-1 field multiplications by keeping the intermediate result in the representation (x, λ_P) .

We now turn to the case of an arbitrary curve $E(\mathbf{F}_{2^n}) = G \times E[2^k]$. Let $P \in G$ and $Q \in \{[\frac{1}{2}]P, [\frac{1}{2}]P + T_2\}$ be given. We want to determine whether $Q = [\frac{1}{2}]P$ or $Q = [\frac{1}{2}]P + T_2$ and we can do this by repeating the procedure in the proof

of Theorem 1. Apply the formulas (i) and (ii) k times: the first time to Q to get a point Q_1 such that $[2]Q_1=Q$. In the i'th step apply the formulas to Q_{i-1} to get a point Q_i such that $[2]Q_i=Q_{i-1}$. The resulting point Q_k will be on the form $[\frac{1}{2^{k+1}}]P+T_{2^{k+1}}$ if and only if $Q=[\frac{1}{2}]P+T_2$ and it will be on the form $[\frac{1}{2^{k+1}}]P+T_{2^i}$ where $0 \le i \le k$ if and only if $Q=[\frac{1}{2}]P$. One thus has the necessary and sufficient condition:

$$Q = \left[\frac{1}{2}\right]P \Leftrightarrow Q_k \in E(\mathbf{F}_{2^n})$$

With the notation $Q=(u,v)=(u,u(u+\lambda_{\lceil\frac{1}{2}\rceil P}))$ and $Q_{k-1}=(u_{k-1},v_{k-1}),\, \lceil\frac{1}{2}\rceil P$ is computed in the following manner: compute u_{k-1}^2 by repeated applications of steps 1,2 and 5 of the Point Halving Algorithm. Use u_{k-1}^2 in the check. If the check is negative put $u:=u+\sqrt{x}$ and put $\lambda_{\lceil\frac{1}{2}\rceil P}:=\lambda_{\lceil\frac{1}{2}\rceil P}+1$. Finally, output $(u,\lambda_{\lceil\frac{1}{2}\rceil P})$. For a general curve the operations to be performed in a point halving which gives the output in the representation $(u,\lambda_{\lceil\frac{1}{2}\rceil P})$ are:

solving k second degree equations

k multiplications

1 check

k or k+1 square roots

3 Computing Efficiently

We show how to perform the check, solve the second degree equation and compute the square root used in the Point Halving Algorithm in an efficient way. By "efficient", we mean time efficient and not necessarily storage efficient. We consider both normal and polynomial bases. In a normal basis everything proceeds smoothly. In a polynomial basis we can likewise perform fast computations, but only if it is possible to store $O(n^2)$ bits.

Normal basis

The results given for the normal basis can be found in [IEEE]. We can view \mathbf{F}_{2^n} as an *n*-dimensional vectorspace over \mathbf{F}_2 . In a normal basis a field element is represented as

$$x = \sum_{i=0}^{n-1} x_i \beta^{2^i} \qquad x_i \in \{0, 1\}$$

where $\beta \in \mathbf{F}_{2^n}$ is chosen such that $\{\beta, \beta^2, \dots, \beta^{2^{n-1}}\}$ is a basis for \mathbf{F}_{2^n} . The normal basis has the feature that computing a square root is done by a left cyclic shift and squaring by a right cyclic shift. The time to compute these operations is negligible.

Assume that the second degree equation $\lambda^2 + \lambda = x$ has solutions in \mathbf{F}_{2^n} . A solution is then given by

$$\lambda = \sum_{i=1}^{n-1} \lambda_i \beta^{2^i}$$
 where $\lambda_i = \sum_{k=1}^i x_k$ for all $1 \le i \le n-1$

We expect the time needed to compute this to be negligible compared to the time needed to compute a field multiplication or an inversion.

Since the time to compute a solution to a second degree equation is negligible we can compute the check in the following way: Compute a candidate λ from x and check if $\lambda^2 + \lambda = x$. If this is not the case then the equation has no solutions in \mathbf{F}_{2^n} .

Polynomial Basis

We will use the representation:

$$x = \sum_{i=0}^{n-1} x_i T^i \qquad x_i \in \{0, 1\}$$

The square root of x can be computed with the storage of the element \sqrt{T} after making the following observations:

– in characteristic two the square root map is a field morphism. – $\sqrt{\sum_{i \ even} x_i T^i} = \sum_{i \ even} x_i T^{\frac{i}{2}}$

$$-\sqrt{\sum_{i \ even} x_i T^i} = \sum_{i \ even} x_i T^{\frac{i}{2}}$$

Now, splitting x into even and odd powers and taking the square root, we get

$$\sqrt{x} = \sum_{i \text{ even}} x_i T^{\frac{i}{2}} + \sqrt{T} \sum_{i \text{ odd}} x_i T^{\frac{i-1}{2}}$$

so all we have to do to compute a square root is to "shrink" two vectors to half size and then perform a multiplication of a precomputed value with an element of length $\frac{n}{2}$. Therefore we expect the time to compute a square root in a polynomial basis to be equivalent to half the time to compute a field multiplication plus a very small overhead.

To perform the check and to solve the second degree equation we will view \mathbf{F}_{2^n} as an *n*-dimensional vectorspace over \mathbf{F}_2 . The map

$$F: \mathbf{F}_{2^n} \to \mathbf{F}_{2^n}$$

 $\lambda \longmapsto \lambda^2 + \lambda$

is then a linear operator with kernel $\{0,1\}$.

For a given x, the equation $\lambda^2 + \lambda = x$ has solutions in \mathbf{F}_{2^n} if and only if the vector x is in the image of F. Im(F) is an n-1 dimensional subspace of \mathbf{F}_{2^n} . For a given basis of \mathbf{F}_{2^n} with corresponding dot product there is a unique non-zero vector which is orthogonal to all vectors in Im(F). Denote this vector w. We then have:

$$\exists \lambda \in \mathbf{F}_{2^n}: \ \lambda^2 + \lambda = x \iff x \bullet w = 0$$

so the check can be performed by adding up the entries of x for which the corresponding entries of w hold a 1. We expect the time to perform the check to be negligible.

To solve the second degree equation $F(\lambda) = \lambda^2 + \lambda = x$ in a polynomial basis we propose a straightforward method which requires capability to store an $n \times n$ matrix. We are looking for a linear operator G such that

$$\forall x \in Im(F): F(G(x)) = (G(x))^2 + G(x) = x$$

Let $\alpha \in \mathbf{F}_{2^n}$ be any vector such that $\alpha \not\in Im(F)$ and define G by

$$G:=\widetilde{F}^{-1} \quad \text{ where } \quad \widetilde{F}(T^i)=\left\{ \begin{matrix} \alpha & \text{when } i=0 \\ F(T^i) & \text{when } 1 \leq i \leq n-1 \end{matrix} \right.$$

With $x = \sum_{i=1}^{n-1} x_i F(T^i) \in Im(F)$ given it is left to the reader to verify that G(x) solves the second degree equation. In an implementation one precomputes the matrix representation for G in the basis $\{1, T, \dots, T^{n-1}\}$. In characteristic two, multiplication of a matrix by a vector is just adding up the columns of the matrix for which the corresponding entries of the vector hold a 1. Therefore, this method for solving a second degree equation on average requires $\frac{n}{2}$ field additions.

The drawback of the method is the storage needed. In appendix B, an algorithm is given which reduces the storage needed to $\frac{n^2}{2}$ bits. It is even faster requiring on average $\frac{n}{4}$ field additions and a small overhead.

We have one further remark on the storage before ending the section. We do not need to store the vector w needed for the check seperately, since it is the first row of the matrix representation of G. It follows from the fact that G is invertible and $G(F(T^i)) = T^i$ for $1 \le i \le n-1$. This implies that the first row of the matrix representation of G is non-zero and orthogonal to all column vectors of the matrix representation of F.

4 Applications for Scalar Multiplication

Let a point $P \in E(\mathbf{F}_{2^n})$ of odd order r and an integer c be given. Let m denote the integer part of $\log_2(r)$. We want to compute the scalar multiple [c]P employing the halving map. For this purpose, we prove the easy:

Lemma 1. For every integer c, there is a rational number of the form

$$\sum_{i=0}^{m} \frac{c_i}{2^i} \quad c_i \in \{0, 1\}$$

such that

$$c \equiv \sum_{i=0}^{m} \frac{c_i}{2^i} \pmod{r}$$

Proof. Calculate the remainder of $2^m c$ after division by r and write the result as a binary number:

$$2^m c \pmod{r} = \sum_{i=0}^m \widehat{c}_i 2^i \quad \widehat{c}_i \in \{0, 1\}$$

Dividing by 2^m and putting $c_i := \widehat{c}_{m-i}$ gives the result:

$$\sum_{i=0}^{m} \frac{c_i}{2^i} := \sum_{i=0}^{m} \widehat{c}_i 2^{i-m} \quad c_i \in \{0, 1\}$$

Let $\langle P \rangle$ denote the cyclic group generated by P. Since we have the isomorphism of rings:

$$< P > \simeq \mathbf{Z}/r\mathbf{Z}$$

 $[k] P \mapsto k$

we can compute the scalar multiple by

$$[c]P = \sum_{i=0}^{m} \left[\frac{c_i}{2^i}\right]P$$

using point halvings and point additions. The well known double-and-add algorithm can be used for the computations. We only have to replace doublings by halvings in this algorithm. One has to perform $\log_2(r)$ halvings and on average $\frac{1}{2}\log_2(r)$ additions. There are improvements to the double-and-add algorithm which require only $\frac{1}{3}\log_2(r)$ additions in the average case. In appendix C, we give an automaton suitable for the halve-and-add algorithm. In general, any method which is based on manipulating an integer represented by its binary expansion should be easy to modify so as to make the same manipulations on the rational number from Lemma 1 represented by its $\frac{1}{2}$ -adic expansion. We bear in mind here in particular the sliding window method.

For the addition of our original point P and the intermediate result Q, we use the following algorithm which is a small modification of the usual addition algorithm given in for example [IEEE]:

Addition Algorithm

Input: P = (x, y) in affine coordinates and $Q = (u, u(u + \lambda_Q))$ represented as

Output:P + Q = (s, t) in affine coordinates

- 1 Compute $\lambda := \frac{y+u(u+\lambda_Q)}{x+u}$ 2 Compute $s := \lambda^2 + \lambda + a + x + u$
- 3 Compute $t := (s+x)\lambda + s + u$
- 4 Output (s,t)

The algorithm requires 1 inversion, 3 multiplications and 1 squaring.

5 Expected Performance

We will only consider curves with minimal two-torsion in this section. The time saved by using halvings instead of doublings is significant. In affine coordinates, both elliptic doubling and addition require 1 inversion, 2 multiplications and 1 squaring. If the scalar for the scalar multiplication is represented by a bitvector of length m with k non-zero entries the operations needed for the scalar multiplication are:

operation	double-and-add	halve-and-add
inversions	m+k	k
multiplications	2m+2k	m+3k
squarings	m+k	k
solving $\lambda^2 + \lambda = a + x$	0	m
square roots	0	m
checks	0	m

Thus, by using halvings one saves m inversions, m-k multiplications and m squarings at the cost of solving m second degree equations, calculating m square roots and performing m checks. We have shown how to compute the "new" operations fast. In the average case of the optimized version of the double-and-add algorithm and halve-and-add algorithm given in Appendix $\mathbb C$ we have $k=\frac{m}{3}$. In a polynomial basis, it is difficult to give a general estimate of the improvement in running time because of the many different operations involved. Based on [SOOS] we will put the time to compute an inversion equivalent to the time to compute 3 multiplications. A field multiplication in the average case requires $\frac{n}{2}$ field additions and afterwards a reduction by the reduction polynomial defining the field. With the following assumptions on equivalence of timings:

1 inversion \sim 3 multiplications

1 multiplication ~ 10 squarings

 $1(\lambda^2 + \lambda = a + x) + 1$ check + 1 square root ~ 1 multiplication + 1 squaring

we get an improvement in the running time on 39%. In a normal basis, as mentioned in Section 3, we assume that the time needed to calculate the square root, the check and the second degree equation is negligible compared to the time needed to compute a multiplication or an inversion. An inversion can be computed using $[\log_2(n-1)] + \omega(n-1) - 1$ multiplications ([Menezes]), where n is the degree of the field extension and ω is the number of 1's in the binary expansion of n-1. As an example, with n=155 the number of multiplications needed for one inversion is 10. The improvement in running time is then 67%. Even with the time to compute an inversion being equivalent to the time to calculate 3 multiplications we get a 55% improvement of the running time.

6 Conclusion

A fast method for elliptic scalar multiplication has been introduced. In its fastest version it applies to half the curves: the ones with minimal two-torsion. For a polynomial basis, the disadvantage is the amount of storage needed. For a normal basis, there are no disadvantages. The algorithm is clearly superior to any double-and-add algorithm when this is implemented using affine coordinates. In [CLNZ] it is investigated how to reduce the amount of curve additions by representing the scalar by a bit vector which is longer, but has a lower Hamming-weight. This is of particular interest in this context since point halving is much faster than point addition. The current limitations of the method give rise to the challenges: Find a fast check for curves with higher two-torsion. Derive an efficient halving algorithm for projective coordinates. Reduce the storage needed in a polynomial basis.

7 Acknowledgements

The author would like to thank Kristian Pedersen, Jean-Bernard Fischer and Jacques Stern for fruitful comments.

References

- MorOli. F. Morain and J. Olivos: Speeding up computations on an elliptic curve using addition-subtraction chains ln Theoretical Informatics and Applications 24, No. 6, 1990 pp.531-544 149
- Zhang. C.N.Zhang: An improved binary algorithm for RSA In Computers and Mathematics with Applications, vol. 25, 1993, pp.15-24 149
- IEEE. Standard Specifications for Public Key Cryptography, Annex A. Number Theoretic Background. IEEE Standards Department, August 20, 1998. 137, 140, 143
- Koblitz. N.Koblitz: CM-Curves with Good Cryptographic Properties, Advances in Cryptology-CRYPTO 91, Lecture Notes in Computer Science, No. 576, Springer-Verlag, Berlin, 1992, pp. 279-287. 135
- Meistaff. W.Meier, O.Staffelbach: Efficient Multiplication on Certain Nonsupersingular Elliptic Curves, Advances in Cryptology-CRYPTO 92, Lecture Notes in Computer Science, No. 740, Springer-Verlag, Berlin, 1992, pp. 333-344. 135
- Muller1. Volker Muller: Fast Multiplication on Elliptic Curves over Small Fields of Characteristic Two, *Journal of Cryptology* 1998, pp. 219-234. 135
- Silverman. J.Silverman: The arithmetic of Elliptic Curves, Graduate Texts in Mathematics 106, Springer-Verlag, Berlin Heidelberg New York 1986. 135
- CLNZ. G.Cohen, A.Lobstein, D.Naccache, G.Zemor: How to Improve an Exponetiation Black-box, Technical Report AP03-1998, Gemplus' Corporate Product R&D Division 145
- Muller2. Volker Muller: Efficient Algorithms for Multiplication on Elliptic Curves TI-9/97,1997, Institut fur theoretische Informatik 149
- Menezes. Alfred J. Menezes: Elliptic Curve Public Key Cryptosystems Kluwer Acedemic Publishers 144
- SOOS. R. Schroeppel, H. Orman, S. O'Malley, O. Spatscheck: Fast Key Exchange with Elliptic Curve Systems, Advances in cryptology - CRYPTO '95, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995 144

A Half the Curves Have Minimal Two-Torsion

Theorem 2. Let a field \mathbf{F}_{2^n} be given. Half the curves defined on \mathbf{F}_{2^n} have minimal two-torsion.

Proof. As mentioned in the introduction, a non-supersingular curve E is defined by an equation

$$y^2 + xy = x^3 + ax^2 + b$$
 $a, b \in \mathbf{F}_{2^n}$ $b \neq 0$

That is, it is defined by the pair $(a,b) \in \mathbf{F}_{2^n} \times \mathbf{F}_{2^n}$. The unique point of order two is given by $T_2 = (0, \sqrt{b})$. With T_2 as input, we can calculate the two points of order four by the equations (i), (ii), (iii) in Section 2. Therefore, by repeating the arguments of Theorem 1 and the analysis afterwards leading to the check in the Point Halving Algorithm, we have with T_2 as input in equation (i):

$$T_4, [3]T_4 \in E(\mathbf{F}_{2^n}) \Leftrightarrow \exists \lambda \in \mathbf{F}_{2^n} : \lambda^2 + \lambda = a$$

Let F denote the linear operator $F(\lambda) = \lambda^2 + \lambda$ with domain \mathbf{F}_{2^n} . We negate the necessary and sufficient condition and get:

E has minimal two-torsion
$$\Leftrightarrow a \not\in Im(F)$$

Since F has kernel $\{0,1\}$, this condition holds for 2^{n-1} values of a, thus half the curves.

B Reducing Storage in a Polynomial Basis

Assume that the equation $F(\lambda) = \lambda^2 + \lambda = x$ has solutions in \mathbf{F}_{2^n} . As explained in Chapter 3, the solutions can be computed with the storage of an $n \times n$ matrix representation of a linear operator G, where G(x) is a solution to the second degree equation. In this Appendix we show how to reduce the storage needed. The idea is to write x as

$$x = F(y) + z$$

where $y = \sum_{i=1}^{n-1} y_i T^i$ and where z is an element of the subspace of \mathbf{F}_{2^n} generated by the vector 1 and the vectors $\{T^i\}$ where i is odd. Define \widetilde{G} by

$$\widetilde{G}(T^i) = \begin{cases} G(1) & \text{if } i = 0 \\ G(T^i) & \text{if } i = 1, 3, 5, \dots \\ 0 & \text{if } i = 2, 4, 6, \dots \end{cases}$$

Then $\widetilde{G}(z) = G(z)$. It follows from the definition of G that $G(F(T^i)) = T^i$ for all $1 \le i \le n-1$. Therefore a solution to the second degree equation is given by

$$G(x) = G(F(y) + z) = G(\sum_{i=1}^{n-1} y_i F(T^i)) + G(z) = y + G(z) = y + \widetilde{G}(z)$$

So, using this approach, we only have to store the $\left[\frac{n}{2}\right] + 1$ nontrivial vectors of the matrix representation of \widetilde{G} . Let x be represented as

$$x = \sum_{i=0}^{n-1} x_i T^i$$

and assume for simplicity that n-1 is a power of two. The following algorithm calculates a solution to the second degree equation using $\log_2(n-1)$ iterations.

Theorem 3. The algorithm works.

Proof. Define values

$$a_k := \begin{cases} \frac{n-1}{2^k} & \text{when } 0 \le k \le \log_2(n-1) \\ 0 & \text{when } k = 1 + \log_2(n-1) \end{cases}$$
$$x^{(0)} := x$$
$$y^{(0)} := 0$$

and recursively for $1 \le k \le \log_2(n-1)$:

$$x^{(k)} = x^{(k-1)} + F(\sum_{i=1+a_{k+1}}^{a_k} x_{2i}^{(k-1)} T^i)$$
$$y^{(k)} = y^{(k-1)} + \sum_{i=1+a_{k+1}}^{a_k} x_{2i}^{(k-1)} T^i$$

corresponding to the operations performed in the algorithm. Using the fact that F is a linear operator, it is easily seen that we have for all $0 \le k \le \log_2(n-1)$:

$$x = F(y^{(k)}) + x^{(k)}$$

In particular, this is true for $k := \log_2(n-1)$. It is immediate from the recursion formula defining $y^{(k)}$ that the constant term of $y^{(\log_2(n-1))}$ is zero. It remains to show that all even coefficients of $x^{(\log_2(n-1))}$ of index greater than or equal to two are zero. Then, a solution can be calculated by $y^{(\log_2(n-1))} + \widetilde{G}(x^{(\log_2(n-1))})$ which is the final step of the algorithm. With $2+a_k \le n-1$ we show by induction on k that

$$x_{2+a_k}^{(k)} = x_{4+a_k}^{(k)} = \dots = x_{n-1}^{(k)} = 0$$
 for all $0 \le k < \log_2(n-1)$

For k = 0 this is trivially true. Assume that the statement holds for k - 1. We then have:

$$\begin{split} x^{(k)} &= x^{(k-1)} + \sum_{i=1+a_{k+1}}^{a_k} x_{2i}^{(k-1)} T^{2i} + \sum_{i=1+a_{k+1}}^{a_k} x_{2i}^{(k-1)} T^i \\ &= \left(x^{(k-1)} + \sum_{i \ even, \ i=2+a_k}^{a_{k-1}} x_i^{(k-1)} T^i \right) + \sum_{i=1+a_{k+1}}^{a_k} x_{2i}^{(k-1)} T^i \end{split}$$

It is clear that the coefficients of the expression in the parenthesis are zero for even indices $2+a_k, \dots, a_{k-1}$. By the induction assumption, the same is true for the even indices $2+a_{k-1}, \dots, n-1$. Adding the term outside the parenthesis does not affect basis vectors of index greater than a_k and this completes the induction. Finally, for the last iteration in the algorithm with $k = \log_2(n-1)$, we get:

$$x^{(k)} = (x^{(k-1)} + x_2^{(k-1)}T^2) + x_2^{(k-1)}T$$

from which we see that $x_2^{(\log_2(n-1))}=0$ and we are done. \Box

Each iteration in the algorithm is fast. The kth iteration consists in removing the relevant coefficients of even index from x, "squeezing" them to a vector of length $\frac{n-1}{2^{k+1}}$ and then add this vector to x and y. It is left to the reader to see that the addition in the reassigning of y is not really an addition but a concatenation. Therefore the total amount of field additions in the loop corresponds to the addition of a vector of length $\frac{n}{2}$. We can thus expect the running time of the algorithm to be a small overhead plus, on average, $\frac{n}{4}$ field additions from the final multiplication of the matrix representation of \widetilde{G} with x.

Using the same idea, one can hope to get the amount of storage even further down by exploiting the specific properties of the reduction polynomial. More precisely, the idea is that one can avoid to store a column vector of high index k if the degree of T^{2k} after reduction by the reduction polynomial is less than k. We can then once more write $x = (x + F(x_k T^k)) + F(x_k T^k)$ and calculate a solution by $G(x) = G(x + F(x_k T^k)) + x_k T^k$ which does not involve computing $G(T^k)$. We will not pursue this further.

C An Optimized Version of the Halving-and-add Algorithm

We give below an automaton which takes as input a point P of odd order and a bitvector (c_0, \dots, c_m) and outputs $\sum_{i=0}^m \left[\frac{c_i}{2^i}\right] P$. The basic idea, which we have from [MorOli], is to minimize the number of curve additions by applying the following identities of strings to the bitvector:

$$\underbrace{1\cdots 1}_{k} = 1\underbrace{0\cdots 0}_{k-1} - 1$$

and

$$\underbrace{1\cdots 1}_{k_1} \underbrace{0} \underbrace{1\cdots 1}_{k_2} = \underbrace{1\cdots 1}_{k_1+1} \underbrace{0\cdots 0}_{k_2-1} -1 = \underbrace{1} \underbrace{0\cdots 0}_{k_1} -1 \underbrace{0\cdots 0}_{k_2-1} -1$$

In this way one can always obtain an identity of rational numbers:

$$\sum_{i=1}^{m} \frac{c_i}{2^i} = \sum_{i=0}^{m} \frac{d_i}{2^i}, \quad d_i \in \{0, \pm 1\}$$

where the coefficients d_i have the further feature that $\forall i \in \{0, \dots, m-1\}: d_i \neq 0 \Rightarrow d_{i+1} = 0$. It is proven in for example [Zhang] that the amount of non-zero coefficients d_i on average will be one third of m. Since the time to calculate -P from P is negligible, we can now calculate

$$\sum_{i=0}^{m} \left[\frac{c_i}{2^i}\right] P = [c_0] P + \sum_{i=1}^{m} \left[\frac{c_i}{2^i}\right] P = [c_0] P + \sum_{i=0}^{m} \left[\frac{d_i}{2^i}\right] P$$

using fewer curve additions and still m halvings. In [Muller2] is given an automaton to be used in an optimization of the Double-and-add Algorithm. It is identical to the automaton given below except for the last bit. For the correctness of the automaton given here we therefore refer to [Muller2]. The automaton is to be used on the bits c_m, \dots, c_0 in descending order of indices. The arrows pointing out are for the calculations regarding the final bit c_0 .

On the Design of RSA with Short Secret Exponent*

Hung-Min Sun¹, Wu-Chuan Yang² and Chi-Sung Laih²

Department of Computer Science and Information Engineering
National Cheng Kung University; Tainan, Taiwan 701
hmsun@mail.ncku.edu.tw

Department of Electrical Engineering
National Cheng Kung University, Tainan, Taiwan 701
wcyang77@ms32.hinet.net
laihcs@eembox.ee.ncku.edu.tw

Abstract. At Eurocrypt'99, Boneh and Durfee presented a new short secret exponent attack which improves Wiener's bound ($d < N^{0.25}$) up to $d < N^{0.292}$. In this paper we show that it is possible to use a short secret exponent which is below these bounds while not compromising with the security of RSA provided that p and q are differing in size and are large enough to combat factoring algorithms. As an example, the RSA system with d of 192 bits, p of 256 bits, and q of 768 bits is secure against all the existing short secret exponent attacks. Besides, in order to balance and minimize the overall computations between encryption and decryption, we propose a variant of RSA such that both e and d are of the same size, e.g., $\log_2 e \approx \log_2 d \approx 568$ for a 1024-bit RSA modulus. Moreover, a generalization of this variant is presented to design the RSA system with $\log_2 e + \log_2 d \approx \log_2 N + l_k$ where l_k is a predetermined constant, e.g., 112. As an example, we can construct a secure RSA system with p of 256 bits, q of 768 bits, d of 256 bits, and e of 880 bits.

1 Introduction

The RSA public-key cryptosystem was invented by Rivest, Shamir, and Adleman [16] in 1978. Since then, the RSA system has been the most well-known and accepted public key cryptosystem. Usually, the RSA system is deployed in various application systems for providing privacy and/or ensuring authenticity of digital data. Hence many practical issues have been sequentially considered when implementing RSA, e.g., how to reduce the storage for RSA modulus [13,24], how to use short public exponent for reducing the encryption execution time (or signature-verification time) [2-4,8-9], and how to use short secret exponent for reducing the decryption execution time (or signature-generation time) [1,25-26]. In this paper, we are interested in the use of short secret exponent because it is particularly advantageous when there is a large difference in computing power between two communicating devices. For

^{*} This wok was supported in part by the National Science Council, Taiwan, under contract NSC-88-2213-E-324-007 and NSC88-2213-E-006-025.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 150-164, 2000. © Springer-Verlag Berlin Heidelberg 2000

example, it would be desirable for a smart card to have a short secret exponent in order to speedup the decryption or the generation of signatures in the smart card, and for a larger computer to have a short public exponent in order to speedup the encryption or the verification of signatures required in the smart card. We are also interested in the use of balanced and minimized public and secret exponents that the length of both is approximately equal and is as short as possible. The main motivation for this is to provide the requirement of those applications when the computing power between two communicating devices is approximately equal. Particularly, it is advantageous when a sequence of encryptions and decryptions (or signature generations and verifications) are required to run synchronously, i.e., no party is idle between communication. Inspired by the above concept, we are also interested in balancing and minimizing the encryption time and the decryption time when there is a difference in computing power between two communicating devices. It is an intuitive thought that if the computation amount in encryption is heavier, then the computation amount in decryption will be lighter, and vice versa. Therefore there should exists a trade-off between encryption and decryption, e.g., the overall computation amount is constant. Consequently, what we concern is how to reduce the overall computations used in encryption and decryption and how to distribute the overall computations between encryption and decryption. If the distributed computations between encryption and decryption are roughly proportional to the computing power of the two communicating devices, we can balance the encryption time and the decryption time even if there is a difference in computing power between these two devices.

We first describe a simplified version of RSA primitive as follows: Let N=pq be the product of two large primes. If both p and q are 512 bits long, then N is about 1024 bits long. Let e and d be two integers satisfying $ed=1 \mod \phi(N)$, where $\phi(N)=(p-1)(q-1)$ is the Euler totient function of N. Here we call N the RSA modulus, e the public exponent, and d the secret exponent. The public key is the pair (N, e) and the secret key is d. For simplicity, we assume the owner of the secret key is Alice. To provide privacy, one can encrypt a message d into a ciphertext d by: d0 by: d1 mod d2 mod d3, while only Alice can decrypt the ciphertext d2 into the plaintext d3 by: d4 mod d6 mod d7, while one can verify the validity of Alice's signature d5 on d6 mod d7 mod d8 while one can verify the validity of Alice's signature d6 on d7 mod d8 mod d9 mod d9 mod d9 satisfies a predetermined redundancy scheme.

For a fixed modulus size, the RSA encryption or decryption time is roughly proportional to the number of bits in the exponent. To reduce the encryption time (or the signature-verification time), one may wish to use a small public exponent e. The smallest possible value for e is 3. If e=3 is used for encryption, it has been proven to be insecure against some short public exponent attacks [9]. The most powerful attack on short public exponent is due to Coppersmith, Franklin, Patarin and Reiter [4]. Under their attack, the RSA primitive is insecure for all public exponents of length up to around 32 bits. Therefore it is suggested to use public exponents of length more than 32 bits. Note that these short public exponent attacks succeed only in the encryption of the RSA primitive. They cannot work in the RSA with the protection of the standards PKCS#1 v2.0 or IEEE P1363.

On the other hand, to reduce the decryption time (or the signature-generation time), one may also wish to use a small secret exponent d. Unfortunately, based on the

convergents of the continued fraction expansion of a given number, Wiener [26] showed that the RSA system can be totally broken if $d < N^{0.25}$. Verheul and van Tilborg [25] proposed an extension of Wiener's attack that allows the RSA system to be broken when d is a few bits longer than $0.25\log_2 N$. For $d > N^{0.25}$, their attack need do an exhaustive search for about 2t+8 bits, where $t=\log_2(d/N^{0.25})$. If t=20 (which leads to an order of magnitude 2^{48}) is feasible to do an exhaustive search, then the RSA system with $d < 2^{20} N^{0.25}$ is insecure. Thus this gives a 20 bits improvement on Wiener's bound. Recently, based on lattice basis reduction, Boneh and Durfee [1] proposed a new attack on the use of short secret exponent. They improved Wiener's bound up to $d < N^{0.292}$. This gives a 43 bits improvement on Wiener's bound if N is the size of 1024 bits. In general, the use of short secret exponent encounters a more serious security problem than the use of short public exponent.

In this paper, we show that it is possible to use a short secret exponent which is below both Wiener's bound and Boneh and Durfee's bound while not compromising the security of RSA provided that p and q are differing in size and are large enough to combat the factoring algorithms which are based on elliptic curves. As an example, when p is the size of 256 bits and q is the size of 768 bits, d of 192 bits is large enough to combat the existing short secret exponent attacks. In this study of the balanced and minimized public and secret exponents, we propose a secure variant of RSA such that e and d are of the same size, e.g., $\log_2 e \approx \log_2 d \approx 568$ for a 1024-bit RSA modulus. We analyze the security of the proposed RSA variant according to the ways of attacking short secret exponent RSA and conclude that the proposed scheme is secure enough to defeat all the existing short secret exponent attacks. Finally, the trade-off between the length of secret exponent and public exponent is analyzed. We is possible to design secure a RSA $\log_2 e + \log_2 d \approx \log_2 N + l_k$ where l_k is a predetermined constant, e.g., 112. Compared with typical RSA system that e is of the same order of magnitude as N if d is first selected, these variants of RSA have the advantage that the overall computations can be significantly reduced. As an example, we can construct a secure RSA system with p of 256 bits, q of 768 bits, d of 256 bits, and e of 880 bits.

The remainder of this paper is organized as follows. In section 2, we review some well-known attacks on the use of short secret exponent. In section 3, we propose and analyze a construction of RSA system to combat those short secret exponent attacks. In section 4, we present a variant of RSA such that the length of the secret exponent and the public exponent can be balanced and minimized. In section 5, the trade-off between the length of the secret exponent and the length of the public exponent is analyzed. Finally, we conclude this paper in section 6.

2 Overview of Previous Works

Because the security analysis of our schemes is related to Wiener's attack [26], Verheul and van Tilborg's attack [25], and Boneh and Durfee's attack [1] on the use of short secret exponent, here we briefly review these attacks as the background

information for reading this paper. Additionally, we also introduce the basic concept of unbalanced RSA which was proposed by Shamir [19].

2.1 Wiener's Attack and its Extension on Short Secret Exponent

Wiener's attack [26] is based on approximations using continued fractions to find the numerator and denominator of a fraction in polynomial time when a close enough estimate of the fraction is known. He showed the RSA system can be totally broken if the secret exponent with up to approximately one-quarter as many bits as the modulus (both p and q are of the same size). For simplicity, we slightly modify Wiener's attack in the following. Let $ed = k\phi(N)+1$ in a typical RSA system. Hence $\gcd(d, k)=1$. We can rewrite this equation as: ed=k(N-(p+q)+1)+1. Therefore, $|\frac{e}{N}-\frac{k}{d}|=\delta$, where

$$\delta = \frac{k}{d} \frac{p + q - 1 - \frac{1}{k}}{N}.$$
 It is known that for a rational number x such that $|x - \frac{B}{A}| < \frac{1}{2A^2}$, where $\gcd(A, B) = 1$, $\frac{B}{A}$ can be obtained as convergents of the continued fraction expansion of x . For further discussion of continued fractions, we refer the reader to [26]. As pointed out by Pinch [15], if $p < q < 2p$ and $d < \frac{1}{3} N^{0.25}$, then $p + q - 1 < 3\sqrt{N}$ and $k < d < \frac{1}{3} N^{0.25}$. Therefore, $|\frac{e}{N} - \frac{k}{d}| \le \frac{1}{dN^{0.25}} < \frac{1}{3d^2} < \frac{1}{2d^2}$. Thus $\frac{k}{d}$ can be found because $\frac{k}{d}$ is one of the log N convergents of the continued fraction for $\frac{e}{N}$.

The extension of Wiener's attack, proposed by Verheul and van Tilborg [25], basically follows Wiener's approach except that they proposed a more general method to compute the convergents of the continued fraction expansion of the same number as in Wiener's attack up to the point where the denominator of the convergent exceeds approximately $N^{0.25}$. For $d > N^{0.25}$, their attack need do an exhaustive search for about 2t+8 bits, where $t = \log_2(d/N^{0.25})$. Because Verheul and van Tilborg's attack is not directly related to our work, we omit reviewing their attack here.

2.2 Boneh and Durfee's Attack on Short Secret Exponent

Based on solving the small inverse problem, Boneh and Durfee [1] proposed a new attack on short RSA secret exponent, which leads to a tighter bound than the bound proposed by Wiener. They concluded that if $e \approx N$ and $d < N^{0.292}$, then the secret exponent d can be efficiently found. In a typical RSA system, $ed = k\phi(N)+1$. So, ed = k((N+1)-(p+q))+1. Let A = N+1 and s = -(p+q), and t = -k. Then ed + t(A+s)=1. Thus $t(A+s)=1 \pmod{e}$. If both t and t are much smaller than t0, the problem can be viewed as follows: given an integer t1, find an element close to t2 whose inverse modulo t3 is

small. This problem is usually referred as the *small inverse problem*. Let $e \approx N^{\alpha}$ and $d < N^{\beta}$. So far, Boneh and Durfee have showed that if $\beta < \frac{7}{6} - \frac{1}{3}(1 + 6\alpha)^{1/2}$, then the small inverse problem can be solved. Consequently, RSA is insecure whenever $d < N^{0.285}$ (which can be slightly improved up to $N^{0.292}$) if $\alpha = 1$.

2.3 Unbalanced RSA System

It is generally accepted that RSA moduli are composed of two large primes of the same size. Shamir [19] proposed a variant of the typical RSA, called *unbalanced RSA*, that the two primes are widely differing in size, e.g., $\log_2 q = 10 \cdot \log_2 p$. His motivation is to provide higher security without increasing computational cost.

In general, all the existing factoring algorithms to break RSA can be classified into two types: algorithms whose running time depends on the smaller factor p, and algorithms whose running time depends on the size of the modulus N. The fastest factoring algorithm of the first type is based on elliptic curves, and its asymptotic running time is $\exp(O((\log_2 p)^{1/2}(\log_2 \log_2 p)^{1/2}))$. This algorithm is usually referred as the elliptic curve method (ECM). So far, the largest factor that has ever been found in practice with this algorithm is about 53 digits (≈176 bits) long [6]. Therefore, if we choose p to be larger than 256 bits, the elliptic curve method becomes infeasible. The fastest factoring algorithm of the second type is based on the general number field sieve (GNFS), and its asymptotic running time is $\exp(O(\log_2 N)^{1/3}(\log_2 \log_2 N)^{2/3})$). So far, the largest RSA modulus has ever been factored in practice with this algorithm is 140 digits (≈ 465 bits) long [5]. As the fast development of computer techniques and factoring algorithms, it is clear that the standard 512-bit RSA modulus no longer provides adequate security and must be significantly increased. Generally, for a large RSA modulus the GNFS attack is much more efficient than the ECM attack. Therefore, there is no need to increase the sizes of the RSA modulus and its prime factors at the same rate. Note that at the Eurocrypt'99 rump session, Shamir [20] announced his design for a special hardware, called "TWINKLE" device which can execute sieve-based factoring algorithms approximately two to three orders of magnitude as fast as a conventional fast PC. If the device can be implemented efficiently, this new technique will increase the size of factorable numbers by 100 to 200 bits for a GNFS attack.

Note that Gilbert *et al.* [7] pointed out that Shamir's unbalanced RSA suffers from some weaknesses. However, these weaknesses come from decrypting only modulo p (and thus limiting the plaintexts to integers smaller than p). Our schemes proposed in this paper don't suffer from the same weaknesses.

Note that some fast and practical public-key cryptosystems [11,14,23] which rely on the difficulty of factoring numbers of the type p^2q were proposed recently. These cryptosystems also use the same concept of making the factors short but large enough such that an ECM attack is infeasible.

3 RSA with Short Secret Exponent

In this section, we propose an unbalanced RSA system such that the use of short secret exponent in the RSA is still secure against all the existing short secret exponent attacks. We show that when p and q are the size of 256 bits and 768 bits, d of 192 bits is large enough to combat all the existing short secret exponent attacks.

3.1 The Proposed Scheme (Scheme I)

We propose a construction of the unbalanced RSA as follows:

- Step 1. Randomly select a prime p and a prime q (p < q) such that p and N is large enough to make an ECM attack and a GNFS attack infeasible respectively, e.g., p and q are 256 bits and 768 bits long, and therefore N is about 1024 bits long.
- Step 2. Randomly select a short secret exponent d such that $\log_2 d + \log_2 p > \frac{1}{3}\log_2 N$ (see Section 3.4) and $d > 2^{\gamma}p^{0.5}$, where γ is a security parameter, e.g., $\gamma = 64$ and hence d is 192 bits long. Note that it is necessary that γ satisfies the following inequality: $32\alpha \gamma \log_e 2 >> 3(1-\alpha-2\gamma \log_e 2)^2$, where $\alpha \approx \log_e q$. Here \log_e denotes the logarithm with the base e that is the public exponent. We give the details in Section 3.3.
- Step 3. Find e such that $ed=1 \mod \phi(N)$, where $\phi(N)=(p-1)(q-1)$. Generally, e will be the same order of magnitude as $\phi(N)$. Here we assume $e \ge \phi(N)/2 + 1$ (the occurrence probability of this case is 1/2). If not, we repeat Step 2 again.

It is clear that the construction leads to a short secret exponent, e.g., a 192-bit d for a 1024-bit RSA modulus, which is far below the lower bounds proposed by Wiener (256 bits [26]) and by Boneh and Durfee (299 bits [1]). Note that if $\gamma \approx 0.5 \log_2 p$, to our best knowledge, no information can be obtained to break the resulting RSA system until now. The details are explained in Section 3.3. So, the RSA system with p of 256 bits, q of 768 bits, d of 256 bits (due to 128-bit γ) and e of 1024 bits is quite secure.

3.2 Combating Wiener's Attack and its Extension

Because $d > 2^{\gamma} p^{0.5}$, it is clear that $\frac{1}{p} > 2^{2\gamma} \frac{1}{d^2}$. Because $k\phi(N) = ed$ -1, we can obtain $\frac{k}{d} = \frac{e - \frac{1}{d}}{\phi(N)} \ge \frac{\phi(N)}{2} + 1 - \frac{1}{d} \ge \frac{\phi(N)}{\phi(N)} \ge \frac{1}{2}$. Thus $|\frac{e}{N} - \frac{k}{d}| = \frac{k}{d} \frac{p + q - 1 - \frac{1}{k}}{N} > \frac{k}{d} \frac{q}{N} = \frac{k}{d} \frac{1}{p} > \frac{1}{2} 2^{2\gamma} \frac{1}{d^2} = 2^{2\gamma} \frac{1}{2d^2} > \frac{1}{2d^2}$. If γ is adequately large, the value $|\frac{e}{N} - \frac{k}{d}|$ will be far away the value $\frac{1}{2d^2}$. Thus, Wiener's attack doesn't apply to Scheme I.

3.3 Combating Boneh and Durfee's Attack

Following Boneh and Durfee's approach, let A=N+1, s=-(p+q), and t=-k. Thus $t(A+s)=1 \pmod e$. Let $|s| < e^{\alpha}$ and $|t| < e^{\beta}$. The sufficient condition for solving the small inverse problem is: $4\alpha(2\beta+\alpha-1) < 3(1-\beta-\alpha)^2$. Because of the limit of space, we provide the details in the full version of this paper.

In our construction $e \approx N$, $q \approx |s| \approx e^{\alpha}$, $d \approx |k| \approx |t| \approx e^{\beta}$, therefore $p = \frac{N}{q} \approx \frac{e}{e^{\alpha}} \approx e^{1-\alpha}$.

Hence $d \approx 2^{\gamma} p^{0.5} \approx 2^{\gamma} e^{0.5(1-\alpha)}$. Let $2^{\gamma} \approx e^{\gamma'}$, i.e., $\gamma' \approx \gamma \log_e 2$. Therefore, $d \approx e^{\gamma' + 0.5(1-\alpha)}$. Thus, $2\beta \approx 2\gamma' + 1 - \alpha$. So, the sufficient condition for solving the small inverse problem can be reduced into $32\alpha\gamma' < 3(1-\alpha-2\gamma')^2$. Hence, in order to combat Boneh and Durfee's attack, it is necessary that γ is adequately large such that the following inequality holds: $32\alpha\gamma\log_e 2 >> 3(1-\alpha-2\gamma\log_e 2)^2$. As an example, we assume p, q, γ and d are 256 bits, 768 bits, 64 bits, and 192 bits long respectively. Thus $\alpha=0.75$ and $\beta=0.1875$. It is clear that $4\alpha(2\beta+\alpha-1)=0.375 >> 3(1-\beta-\alpha)^2=0.117186$. So, Boneh and Durfee's attack cannot succeed.

An important observation proposed by Boneh and Durfee [1] is that the unique solution of the small inverse problem encodes enough information to find d. Therefore, a strong resistance to Boneh and Durfee's attack is to make the small inverse problem failed to have a unique solution. This is why Boneh and Durfee believed that a typical RSA with $d \approx N^{0.5}$ is strongly secure against short secret exponent attacks. So, if we let γ be slightly larger than $0.5\log_2 p$, then d > p. Without loss of generality, we assume $|t| \approx d > p \approx e^{1-\alpha}$, then $|t| > e^{1-\alpha}$. Thus the resulting small inverse problem: $t(A+s) = 1 \pmod{e}$, where $|t| > e^{1-\alpha}$ and $|s| \approx e^{\alpha}$, will no longer have a unique solution. As a result, if d is a few bits larger than p, the

resulting RSA is strongly secure against Boneh and Durfee's attack even if Boneh and Durfee's attack can be up to $d < N^{0.5}$.

3.4 Combating the Cubic Attack

Here we consider a kind of attack, named the cubic attack, in the following.

Because ed=k(p-1)(q-1)+1 and N=pq, we can obtain the following system of modular equations:

- (1) $k(p-1)(q-1)+1=0 \pmod{e}$
- (2) $pq = N \pmod{e}$.

Combining (1) and (2), we can obtain the following cubic equation in two variables k and p:

(3)
$$k(p-1)(N-p)+p=0 \pmod{e}$$

Coppersmith [2] has shown how to solve such cubic equations heuristically if

$$\log_2 k + \log_2 p < \frac{1}{3} \log_2 e$$
. To combat Coppersmith's attack, we need the constraint:

$$\log_2 d + \log_2 p > \frac{1}{3} \log_2 N$$
 because $\log_2 k \approx \log_2 d$ and $\log_2 e \approx \log_2 N$ in Scheme

I. On the other hand, if one can know the exact value of k, then the equation (3) can be reduced to a quadratic equation in a single variable p and hence can be solved provided that either e is prime, or can be factored and doesn't have too many prime factors. Therefore, we must make k unknown to an attacker. In Scheme I, because k is of the same order of magnitude as d, it is large enough to make an exhaustive search infeasible. Hence Scheme I is secure against the cubic attack.

4 RSA with Balanced Public Exponent and Secret Exponent

Traditionally, when constructing RSA, p and q are first selected. After that, either first select the secret exponent d and then determine the public exponent e, or vice versa. Thus either e or d is of the same order of magnitude as $\phi(N)$. In this section, we are interested in constructing RSA with balanced and minimized public and secret exponents such that both are approximately $(\frac{1}{2}\log_2 N + 56)$ bits long without compromising the security of RSA. Different from traditional constructions, we first select p and q, and then determine q and q.

4.1 The Proposed Scheme (Scheme II)

Theorem 1. Let two integers a, b > 1. If gcd(a, b) = 1, then we can find a unique pair (u_h, v_h) satisfying au_h - bv_h =1, where (h-1)b< u_h < hb and (h-1)a< v_h < ha, for any integer $h \ge 1$.

We assume p and q are approximately $(\frac{1}{2}\log_2 N - 112)$ and $(\frac{1}{2}\log_2 N + 112)$ bits long respectively. Here we assume that p and N are large enough to make an ECM attack and a GNFS attack infeasible, e.g., p and N are about 400 bits and 1024 bits long respectively. Our construction is the following:

- Step 1. Randomly select a prime number p of $(\frac{1}{2}\log_2 N 112)$ bits.
- Step 2. Randomly select a number k of 112 bits.
- Step 3. Randomly select a number d of $(\frac{1}{2}\log_2 N + 56)$ bits such that gcd(k(p-1), d) = 1.
- Step 4. Based on Theorem 1, we can uniquely determine two numbers u' and v' such that du'-k(p-1)v'=1, where 0 < u' < k(p-1) and 0 < v' < d.
- Step 5. If $gcd(v'+1, d) \neq 1$, then go to Step 3.
- Step 6. Randomly select a number h of 56 bits, compute u = u' + hk(p-1) and v = v' + hd.
- Step 7. If v+1 isn't a prime number, then go to Step 6.
- Step 8. Let e=u, q=v+1, and N=pq, then p, q, e, d, and N are the parameters of RSA.

Clearly, in this construction e and d satisfy the equation: ed = k(p-1)(q-1)

1)+1= $k\phi(N)$ +1. Therefore, the equation: $ed=1 \mod \phi(N)$ still holds as that in typical RSA. Obviously, both e and d obtained from this construction are approximately $(\frac{1}{2}\log_2 N + 56)$ bits long, and p and q are approximately $(\frac{1}{2}\log_2 N - 112)$ bits and $(\frac{1}{2}\log_2 N + 112)$ bits long respectively. As an example, if $\log_2 N \approx 1024$, then d is 568 bits long, p is 400 bits long, e is about 568 bits long and q is about 624 long. A concrete example for this case is given in Appendix A. In order to measure the efficiency of the proposed scheme, we ran some experiments to test the average times required to find a suitable h in Step 6 for obtaining a prime q. Under 100 samples, our results indicate that in average we need try 487.48 times for Step 6 when N is of 1024 bits long. A comparative result is 566.31 times of selecting a random number of 624 bits and testing whether the number is a prime. This shows that both have approximately the same cost in order to obtain a prime q. Note that in Step 5, if $gcd(v'+1, d) \neq 1$, it implies that it is impossible to find h such that v'+hd+1 is a prime. In addition, the prime p generated in Step 1 can be arbitrarily determined, e.g., selecting a strong prime p, but the prime q generated in Step 8 cannot. Fortunately, the requirement for RSA key that p and q are strong primes is no longer needed due to [17,21-22].

Note that compared with the RSA with CRT-based implementations, Scheme II apparently doesn't provide better efficiency. However the CRT-based RSA needs to keep more secrets p and q than the typical RSA. Moreover, the CRT-based RSA usually incurs some additional security problems [12], even some error detection techniques are applied to it.

4.2 Combating Wiener's Attack and its Extension:

Here we examine the security of Scheme II following the line of the attack, proposed by Wiener, on short RSA secret exponent.

It is clear that $|\frac{e}{N} - \frac{k}{d}| = \frac{k}{d} \frac{p + q - 1 - \frac{1}{k}}{N} > \frac{k}{d} \frac{q}{N} = \frac{k}{d} \frac{1}{p}$. Without loss of generality, we assume that $k > 2^{111}$, $2^{-113} N^{0.5} and <math>2^{55} N^{0.5} < d < 2^{56} N^{0.5}$. Hence, $\frac{1}{p} > \frac{2^{112}}{N^{0.5}}$ and $\frac{1}{N^{0.5}} > 2^{55} \frac{1}{d}$. Thus $\frac{k}{d} \frac{1}{p} > 2^{111} \frac{1}{d} \frac{2^{112}}{N^{0.5}} > 2^{279} \frac{1}{2d^2} >> \frac{1}{2d^2}$. So, $|\frac{e}{N} - \frac{k}{d}|$ will be much larger than $\frac{1}{2d^2}$. Thus, Wiener's attack doesn't apply to Scheme II.

4.3 Combating Boneh and Durfee's Attack:

4.4 Combating the Cubic Attack

Here we refer to Section 3.4. In Scheme II, because $\log_2 k + \log_2 p = \frac{1}{2}\log_2 N >> \frac{1}{3}\log_2 e$, Coppersmith's attack cannot work here. In addition, because k is 112 bits long, it is large enough to make an exhaustive search infeasible. Hence Scheme II is secure against the cubic attack.

5 Trade-off between Public Exponent and Secret Exponent

From Section 4, we know that it is possible for us to use median public and secret exponents in RSA system such that the overall computations required in encryption and decryption are minimized and balanced without compromising with the security of RSA. Therefore, one may be desirable to have secret and public exponents which are differing in size, but the overall computations are still minimized, e.g., $d \approx N^{0.25}$ and $e \approx N^{0.86}$. To minimize the overall computations required in encryption and decryption, it is natural that there exists a trade-off between the length of the public exponent and the length of the secret exponent. In this section, we are interested in addressing this problem.

5.1 The Proposed Scheme (Scheme III)

In the following, generalizing Scheme II, we give an efficient construction of RSA such that $\log_2 e + \log_2 d \approx \log_2 N + l_k$, where l_k is a predetermined constant.

- Step 1. Randomly select a prime number p of length l_p ($l_p < \frac{1}{2} \log_2 N$) such that it is large enough to make an ECM attack infeasible, e.g., $l_p = 256$.
- Step 2. Randomly select a number k of length l_k , e.g., $l_k = 112$.
- Step 3. Randomly select a number d of length l_d such that gcd(k(p-1), d)=1, e.g., $l_d=256$.
- Step 4. Based on Theorem 1, we can uniquely determine two numbers u' and v' such that du'-k(p-1)v'=1, where 0 < u' < k(p-1) and 0 < v' < d.
- Step 5. If $gcd(v'+1, d) \neq 1$, then go to Step 3.
- Step 6. Randomly select a number h of length $\log N l_p l_d$, compute u = u' + hk(p-1) and v = v' + hd.
- Step 7. If v+1 isn't a prime number, then go to Step 6.
- Step 8. Let e=u, q=v+1, and N=pq, then p, q, e, d, and N are the parameters of RSA.

From Step 1-3, we know that k, p, and d are l_k bits, l_p bits, and l_d bits long. Obviously, e and q obtained from the above construction are roughly $\log N + l_k - l_d$ bits and $\log_2 N - l_p$ bits long. These parameters l_k , l_p , and l_d must satisfy the following requirements:

- (1) $l_k >> l_p l_d + 1$. (See 5.2)
- (2) α and β must satisfy: $4\alpha(2\beta + \alpha 1) >> 3(1 \beta \alpha)^2$, where $\alpha = \frac{\log_2 N l_p}{\log_2 N + l_k l_d}$ and $\beta = \frac{l_k}{\log_2 N + l_k l_d}$. (See 5.3)

(3) k is large enough to make an exhaustive search infeasible and $l_k + l_p > \frac{1}{3} \log_2 N$ (see Section 5.4)

As an example, if k, p, and d are 112 bits, 256 bits, and 256 bits long, then e is about 880 bits long and q is about 768 bits long. A concrete example for this case is given in **Appendix B**. In order to measure the efficiency of the proposed scheme, we also ran some experiments to test the average times required to find a suitable h in Step 6 for obtaining a prime q. Under 100 samples, our results indicate that in average we need try 743.56 times for Step 6. A comparative result is 696.86 times of selecting a random number of 768 bits and testing whether the number is a prime. This shows that both have approximately the same cost in order to obtain a prime q.

Note that if one wish to have a smaller public exponent and a larger secret exponent, he need only modify this construction by interchanging the positions of e and d, i.e., he first fixes e and p and then determines d and q.

5.2 Combating Wiener's Attack and its Extension

Here we refer to Section 3.2. It is clear that $|\frac{e}{N} - \frac{k}{d}| > \frac{k}{d} \frac{1}{p}$. Without loss of generality, we assume that $k > 2^{l_k - 1}$, $2^{l_p - 1} and <math>2^{l_d - 1} < d < 2^{l_d}$. Hence, $\frac{1}{p} > 2^{-l_p}$ and $2^{-l_d + 1} > \frac{1}{d}$. Obviously, $\frac{k}{d} \frac{1}{p} > 2^{l_k - l_p - 1} \frac{1}{d} = 2^{l_k - l_p} \frac{1}{2d}$. From requirement (1): $l_k >> l_p - l_d + 1$, we know that $l_k - l_p >> -l_d + 1$. Therefore, $\frac{k}{d} \frac{1}{p} >> 2^{-l_d + 1} \frac{1}{2d} > \frac{1}{2d^2}$. So, $|\frac{e}{N} - \frac{k}{d}|$ is much larger than $\frac{1}{2d^2}$. Thus, Wiener's attack doesn't apply to Scheme III.

5.3 Combating Boneh and Durfee's Attack

Because k, p, and d are l_k bits, l_p bits, and l_d bits long, e and q obtained from Scheme III will be roughly $\log N + l_k - l_d$ bits and $\log N - l_p$ bits long. Note that q > p. Let $|s| < e^{\alpha}$ and $|t| < e^{\beta}$. Therefore, $\alpha \approx \frac{\log_2 N - l_p}{\log_2 N + l_k - l_d}$ and $\beta \approx \frac{l_k}{\log_2 N + l_k - l_d}$. As described in Section 3.3, to combat Boneh and Durfee's attack, α and β must satisfy: $4\alpha(2\beta + \alpha - 1) >> 3(1 - \beta - \alpha)^2$. As an example, if k, p, and d are 112 bits, 256 bits, and 256 bits long (hence e and q are about 880 bits and

768 bits long), then
$$\alpha \approx \frac{768}{880}$$
 and $\beta \approx \frac{112}{880}$. It is clear that $4\alpha(2\beta + \alpha - 1) = 0.4443 >> 3(1 - \beta - \alpha)^2 = 0$.

5.4 Combating the Cubic Attack

Here we refer to Section 3.4. In Scheme III, because $\log_2 k + \log_2 p > \frac{1}{3} \log_2 N > 0$ $\frac{1}{3}\log_2 e$, Coppersmith's attack cannot work here. Besides, $l_k=112$ makes an exhaustive search infeasible. Therefore, Scheme III is secure against the cubic attack.

6 Conclusions

An important observation obtained in this paper is that making the size of p and q different enhances RSA to combat all the existing short secret exponent attacks. Although this also reduces the strength of RSA against factoring, p of 256 bits is large enough to combat an ECM attack at present. Hence RSA with d of 192 bits, p of 256 bits, and q of 768 bits is secure against Boneh and Durfee's attack. To our best knowledge, d of 256 bits is quite secure even if Boneh and Durfee's attack can be up to d< $N^{0.5}$. We also propose an efficient construction of RSA with $\log_2 e = \log_2 d =$ $\log_2 N + 56$ and its generalization with $\log_2 e + \log_2 d \approx \log_2 N + l_k$ where l_k is a predetermined constant. These two constructions are also secure against all the existing short secret exponent attacks due to making the size of p and q different. As an example, RSA with e of 568 bits, d of 568 bits, p of 400 bits, and q of 624 bits and RSA with p of 256 bits, q of 768 bits, d of 256 bits, and e of 880 bits are both secure.

Remark: After we finished this paper, Marc Joye provided us with a related article by Sakai, Morii, and Kasahara [18]. In the paper, they proposed a key generation algorithm for RSA cryptosystem which can make $\log_2 e + \log_2 d \approx \log_2 N$. Their schemes have the following properties:

(1)
$$ed = \frac{k(p-1)(q-1)}{2g} + 1$$
, where g is a large prime and $g|(p-1)$, $g|(q-1)$

$$(2) \log_2 p \approx \log_2 q \approx \frac{1}{2} \log_2 N$$

It should be noticed that Wiener [26] has pointed out that making g large (and hence GCD(p-1, q-1) is large) may cause some security problems. For example, one can find g from N-1 by factoring algorithms because g divides pq-1=(p-1)(q-1)+(p-1)+(q-1)1). If g isn't large enough to combat an ECM attack, e.g., g of 110 bits and 120 bits in Sakai et al.'s schemes, g can be found from N-1 easily. Even if g is large enough to combat an ECM attack, e.g, 250 bits long, it is still possible to factor N-1 and hence obtain g because N-1 may possibly contain only some small prime factors excluding g. Once g is obtained, Wiener's attack can work efficiently [26]. A possible solution to repair their schemes is to make g much larger (> 250 bits) and let N-1 contain at least two large prime factors (> 250 bits) including g. However, in some literatures and current practical use, e.g., X9.31, it is usually recommended to make GCD(p-1, q-1) small in order to guard against the relevant attacks such as repeat encryption attacks.

Acknowledgments. We are grateful to Marc Joye for providing us with reference [18] and his valuable comments. We also thank Sung-Ming Yen and the anonymous referees for their helpful comments.

References

- 1. D. Boneh and G. Durfee, "Cryptanalysis of RSA with private exponent $d < N^{0.292}$ ", *Proc. of EUROCRYPT'99*, LNCS 1592, Springer-Verlag, pp. 1-23, 1999.
- 2. D. Coppersmith, "Finding a small root of a univariate modular equation", *Proc. of EUROCRYPT'96*, LNCS 1070, Springer-Verlag, pp. 155-165, 1996.
- 3. D. Coppersmith, "Small solutions to polynomial equations, and low exponent RSA vulnerabilities", *Journal of Cryptology*, Vol. 10, pp. 233-260, 1997.
- 4. D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, "Low-exponent RSA with related messages", *Proc. of EUROCRYPT'96*, LNCS 1070, Springer-Verlag, pp. 1-9, 1996.
- 5. S. Cavallar, W. Lioen, H. te Riele, B. Dodson, A. Lenstra, P. Leyland, P.L. Montgomery, B. Murphy, P. Zimmermann, "Factorization of RSA-140 using the Number Field Sieve", Proc. of ASIACRYPT'99, Springer-Verlag, 1999.
- 6. ECMNET Project; http://www.loria.fr/~zimmerma/records/ecmnet.html
- 7. H. Gilbert, D. Gupta, A. Odlyzko, and J.J. Quisquater, "Attacks on Shamir's RSA for paranoids", *Information Processing Letters*, Vol. 68, pp. 197-199, 1998.
- 8. J. Hastad, "On using RSA with low exponent in a public key network", *Proc. of CRYPTO'85*, LNCS, Springer-Verlag, pp. 403-408, 1986.
- 9. J. Hastad, "Solving simultaneous modular equations of low degree", *SIAM J. of Computing*, Vol. 17, pp. 336-341, 1988.
- 10.I.N. Herstein, *Topics in Algebra*, Xerox Corporation, 1975.
- 11.D. Hühnlein, M.J. Jacobson, S. Paulus, and T. Takagi, "A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption", *Proc. of EUROCRYPT'98*, LNCS 1403, Springer-Verlag, pp. 294-307, 1998.
- 12.M. Joye, J.J. Quisquater, S.M. Yen, and M. Yung, "Security paradoxes: how improving a cryptosystem may weaken it", *Proceedings of the Ninth National Conference on Information Security*, Taiwan, pp. 27-32, May 14-15, 1999.
- 13.A. Lenstra, "Generating RSA moduli with a predetermined portion", *Proc. of ASIACRYPT'98*, LNCS 1514, Springer-Verlag, pp. 1-10, 1998.
- 14.T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring", Proc. of EUROCRYPT'98, LNCS 1403, Springer-Verlag, pp. 308-318, 1998.
- 15.R. Pinch, "Extending the Wiener attack to RSA-type cryptosystems", *Electronics Letters*, Vol. 31, No. 20, pp. 1736-1738, 1995
- 16.R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communication of the ACM*, Vol. 21, pp. 120-126, 1978.

- 17.R. Rivest and R. D. Silverman, "Are strong primes needed for RSA?", in The 1997 RSA Laboratories Seminar series, Seminar Proceedings, 1997.
- 18.R. Sakai, M. Morii, and M. Kasahara, "New key generation algorithm for RSA cryptosystem", *IEICE Trans. Fundamentals*, Vol. E77-A, No. 1, pp. 89-97, 1994
- 19.A. Shamir, "RSA for paranoids", CryptoBytes, Vol. 1, No. 3, pp. 1,3-4, 1995.
- 20.A. Shamir, "Factoring large numbers with the TWINKLE device", presented at *Eurocrypt'99*, 1999.
- 21.R. D. Silverman, "Fast generation of random, strong RSA primes", *CryptoBytes*, Vol. 3, No. 1, pp. 9-13, 1997.
- 22.R. D. Silverman, "The requirement for strong primes in RSA", RSA Laboratories Technical Note, May 17, 1997.
- 23.T. Takagi, "Fast RSA-type cryptosystem modulo p^2q ", *Proc. of CRYPTO '98*, LNCS 1462, Springer-Verlag, pp. 318-326, 1998.
- 24.S.A. Vanstone and R.J. Zuccherato, "Short RSA keys and their generation", *Journal of Cryptology*, Vol. 8, pp. 101-114, 1995.
- 25.E. Verheul and H. van Tilborg, "Cryptanalysis of less short RSA secret exponents", *Applicable Algebra in Engineering, Communication and Computing*, Springer-Verlag, Vol. 8, pp. 425-435, 1997.
- 26.M. Wiener, "Cryptanalysis of short RSA secret exponents", *IEEE Transactions on Information Theory*, Vol. 36, No. 3, pp. 553-558, 1990.

Appendix A: An example for p of 400 bits, q of 624 bits, d of 568 bits, and e of 568 bits

- *p*=0000cd0a 73cb74b6 27aa29e7 9b1a3c1b d73f4b67 92abde25 c2dcc2dd 68f7a477 9cc6f0a0 d5eeea7c 7c740c8c b370a2e1 6112a393
- *q*=0000807e 4aac7213 62d7d547 4e4dac07 1ea03096 0f13c597 a619a6d7 4c8a3e5b dcd00bcb dcfb0758 555f6b4e 23cc4f6a 5221fa87 bfef172d b815a296 4c5c5be7 61a22fe4 53808fac 0a2fb2d2 548285af
- *d*=008dc2d0 0c1e3027 e0a43f18 022896a0 35379c76 b1e5577c 71038464 bf9ef9a6 00bb3aa0 bb4f590d ef8311ab 95282426 7277f349 200c5d67 5e23dc05 9613dccc ae0a5dad 1209cc53
- *e*=00b335b3 9edd0f90 546f4a51 2ec2a0dd 191e1fb0 38f6b5dd b9ef5156 7ecdc538 355a67b6 d7fbbee3 0926925c b0112914 bbe9f4bf a1a61f92 53dfab7e d9c40261 6fc3d7a8 f77c025f

Appendix B: An example for p of 256 bits, q of 768 bits, d of 256 bits, and e of 880 bits

p=f80dd4da c85afbd9 019d0f24 92c03006 c5baef83 7cfc15eb 2e17b1c1 1fb166e3
q=96d12784 058456cf 00e17f03 b6402825 00a95a1a 772f7059 ea78ac03 57e49dbf feaff1d1 b556e47f 855e8d74 9905753b 12a46068 ce6df746 0e85602c 8f4ed8ac ed6b7f21 2fb1d58f ca645447 ae39277d d01e681a e8a630c6 8c158859 c2e4b743
d=bd82175c 6d9bd203 9ce3f83b cdbceb8e 51c82b29 7f4e237d b0eb3518 807c02bf
e=0000c3b8 1c856425 ff98f54d 605ebe3e 58fd6381 acd328b8 0c4c1d7d ebba6832 061d6fa7 baa8b814 65a82be5 93cdc56a 21ac87e7 693e97e9 3632dfc7 47572a58 f3683163 cd312935 bd24a7ac 08204830 1ba73867 da7456d7 f5efcada 715ad9a0 cec3edd3 e773421b 2c699c42 ef62ebff

Efficient Public-Key Cryptosystems Provably Secure against Active Adversaries

Pascal Paillier¹ and David Pointcheval²

Gemplus Cryptography Department, 34 Rue Guynemer, 92447 Issy-Les-Moulineaux, France and ENST, 46 Rue Barrault, 75634 Paris Cedex 13, France. Pascal.Paillier@gemplus.com

 2 LIENS – CNRS, École Normale Supérieure, 45 rue d'Ulm, 75230 Paris Cedex 05, France.

David.Pointcheval@ens.fr, http://www.dmi.ens.fr/~pointche.

Abstract. This paper proposes two new public-key cryptosystems semantically secure against adaptive chosen-ciphertext attacks. Inspired from a recently discovered trapdoor technique based on composite-degree residues, our converted encryption schemes are proven, in the random oracle model, secure against active adversaries (NM-CCA2) under the assumptions that the Decision Composite Residuosity and Decision Partial Discrete Logarithms problems are intractable. We make use of specific techniques that differ from Bellare-Rogaway or Fujisaki-Okamoto conversion methods. Our second scheme is specifically designed to be efficient for decryption and could provide an elegant alternative to OAEP.

1 Introduction

Diffie and Hellman's famous paper [7] initiated the paradigm of asymmetric cryptography in the late seventies but since, very few trapdoor mechanisms were found that fulfill satisfactory security properties. Of course, the first security criterion a cryptosystem has to verify is the one-wayness of its encryption function, but this notion does not suffice to evaluate (and get people convinced of) the strength of an encryption scheme.

A typical example is RSA [18] which, although very popular and widely used in many cryptographic applications, suffers from being *malleable* and consequently requires an additional treatment (some probabilistic padding) on the plaintext in order to strengthen its practical security. Resistance against chosen-ciphertext attacks, in this case, relies on the conjoint use of an external paradigm instead of being inherently provided, although this empirical approach may sometimes appear insufficient, as shown by Bleichenbacher [4] and more recently by Coron, Naccache and Stern [5]. This motivates the construction of provably secure padding techniques such as OAEP [3] or Fujisaki-Okamoto [10].

Considerable efforts have recently been made to investigate cryptosystems achieving provable security against active adversaries at reasonable encryption and/or decryption cost. Our paper introduces two such cryptosystems that are

efficient for decryption and meet provable security at the strongest level (NM-CCA2) in the random oracle model. We make use of specific techniques that differ from those of [3] and [10].

We begin by briefly surveying known notions of security for public-key encryption schemes, referring the reader to [1] for their formal definitions and connections.

1.1 Notions of Security

Formalizing another security criterion that an encryption scheme should verify beyond one-wayness, Goldwasser and Micali [11] introduced the notion of semantic security. Also called indistinguishability of encryptions (or IND for short), this property captures the idea according to which an adversary should not be able to learn any information whatsoever about a plaintext, its length excepted, given its encryption. The property of non-malleability (NM), independently proposed by Dolev, Dwork and Naor [8], supposes that, given the encryption of a plaintext x, the attacker cannot produce the encryption of a related plaintext x'. Here, rather than learning some information about x, the adversary will try to output the encryption of x'. These two properties are related in the sense that non-malleability implies semantic security for any adversary model, as pointed out in [8] and [1].

On the other hand, there exist several types of adversaries, or attack models. In a chosen-plaintext attack (CPA), the adversary has access to an encryption oracle, hence to the encryption of any plaintext she wants. Clearly, in a public-key setting, this scenario cannot be avoided. Naor and Yung [13] consider non-adaptive chosen-ciphertext attacks (CCA1) (also known as lunchtime or midnight attacks), wherein the adversary gets, in addition, access to a decryption oracle before being given the challenge ciphertext. Finally, Rackoff and Simon [17] defined adaptive chosen-ciphertext attacks (CCA2) as a scenario in which the adversary queries the decryption oracle before and after being challenged; her only restriction here is that she may not feed the oracle with the challenge ciphertext itself. This is the strongest known attack scenario.

Various security levels are then defined by pairing each goal (IND or NM) with an attack model (CPA, CCA1 or CCA2), these two caracteristics being considered separately. Interestingly, it has been shown that IND-CCA2 and NM-CCA2 were strictly equivalent notions [1].

Beyond this, Bellare and Rogaway [3] proposed the concept of *plaintext* awareness, where the adversary attempts to produce a valid ciphertext without knowing the corresponding plaintext. This additional security notion was only properly defined in the random oracle model [2].

1.2 The Random Oracle Model

The random oracle model was proposed by Bellare and Rogaway [2] to provide heuristic (yet satisfactorily convincing) proofs of security. In this model, hash functions are considered to be ideal, i.e. perfectly random. From a security

viewpoint, this impacts all three adversary models by giving the attacker an additional access to the random oracles of the scheme.

1.3 Related Work

The basic El Gamal encryption scheme [9], which one-wayness relates to the celebrated Diffie-Hellman (DH) problem, was recently proven semantically secure (i.e. secure in the sense of IND-CPA) by Tsiounis and Yung [22] under the Decision Diffie-Hellman (D-DH) assumption. However, just like RSA, the original scheme remains totally unsecure regarding active attacks. The same authors therefore proposed a converted scheme provably secure in the sense of NM-CCA2 in the random oracle model, under the D-DH assumption in addition to a non-standard one. Independently, Shoup and Gennaro [20] proposed another converted scheme NM-CCA2 in the random oracle model under the D-DH assumption only. The same year, Cramer and Shoup [6] also presented an El Gamal-based cryptosystem, the first to be simultaneously pratical and provably NM-CCA2 secure in the standard model, provided that the D-DH assumption holds.

Several authors have investigated other intractability assumptions. Point-cheval [16] proposed DRSA, an encryption scheme based on the Dependent-RSA Problem, and provided efficient variants provably NM-CCA2 secure in the random oracle model under the hypothesis that the decisional version of the D-RSA Problem is intractable. Naccache and Stern [12], and independently Okamoto and Uchiyama [14] investigated different approaches based on high degree residues. The one-wayness (resp. semantic security) of their schemes is ensured by the Prime Residuosity assumption (resp. the hardness of distinguishing prime-degree residues). Finally, Paillier [15] proposed an encryption scheme based on composite-degree residues wherein semantic security relies on a similar assumption (see below).

In 94, Bellare and Rogaway [3] proposed OAEP, a specific hash-based treatment applicable to any one-way trapdoor permutation to make it secure in the sense of NM-CCA2. Standing in the random oracle model, their security proof is widely recognized and initiated the upcoming RSA-based PKCS #1 V2.0 standard [19]. More recently, Fujisaki and Okamoto [10] discovered a generic conversion method which transforms any semantically secure encryption scheme into a scheme secure in the sense of NM-CCA2 in the random oracle model. The conversion is low-cost for encryption (one additional hash), but appears to be heavy for decryption¹.

1.4 Outline of the Paper

In this paper, we propose two new encryption schemes that are provably secure against adaptive chosen-ciphertext attacks (NM-CCA2) in the random oracle

 $^{^{1}}$ the converted decryption process includes a complete data re-encryption.

model. Based on Paillier's probabilistic encryption schemes [15], we provide semantic security relatively to two number-theoretic decisional problems, namely the Decision Composite Residuosity and Decision Partial Discrete Log problems. With an efficiency comparable to OAEP for decryption, we believe that the second of these cryptosystems could provide an elegant alternative to the new standard.

2 The Basic Schemes

This section briefly describes the public-key cryptosystems proposed in [15], keeping the same notations as in the original paper.

2.1 Notations

We set n=pq where p and q are large primes. We will denote by ϕ Euler's totient function and by λ Carmichael's function on n, i.e. $\phi=(p-1)(q-1)$ and $\lambda=\text{lcm}(p-1,q-1)$ in the present case. For technical reasons, we will focus on moduli n=pq such that $\gcd(p-1,q-1)=2$, which yields $\phi=2\lambda$. Recall that $|\mathbb{Z}_{n^2}^*|=\phi(n^2)=n\phi$ and that Carmichael's theorem implies that

$$\forall w \in \mathbb{Z}_{n^2}^*, \quad \begin{cases} w^{\lambda} = 1 \mod n \\ w^{n\lambda} = 1 \mod n^2 \end{cases}$$

We denote by RSA [n, e] the well-known problem of extracting e-th roots modulo n where n = pq is of unknown factorization.

2.2 Setting

Let n=pq be a modulus chosen as above and $g\in\mathbb{Z}_{n^2}^*$. It is known that the integer-valued function \mathcal{E}_g defined as

$$\begin{array}{cccc} \mathbb{Z}_n \times \mathbb{Z}_n^* & \longrightarrow \mathbb{Z}_{n^2}^* \\ (x,y) & \longmapsto -y^x \cdot y^n \bmod n^2 \end{array}$$

is a bijection if the order of g in $\mathbb{Z}_{n^2}^*$ is a multiple of n. When this condition is met, then given $w \in \mathbb{Z}_{n^2}^*$, the unique integer x for which there exists a y such that $\mathcal{E}_g(x,y)=w$ is called the (n-residuosity) class of w and is denoted $\llbracket w \rrbracket_g$. It is believed that for given n, g and w the problem of computing the class $\llbracket w \rrbracket_g$ of w is computationally hard: this is known as the composite residuosity assumption (CRA) [15].

It has been shown, however, that the knowledge of the factors p and q is sufficient for computing the class of any integer $w \in \mathbb{Z}_{n^2}^*$. Indeed, setting

$$\mathcal{S}_n = \{ u < n^2 \mid u = 1 \bmod n \} ,$$

and

$$\forall u \in \mathcal{S}_n \quad L(u) = (u-1)/n ,$$

we have

$$\llbracket w \rrbracket_g = \frac{L(w^{\lambda} \bmod n^2)}{L(g^{\lambda} \bmod n^2)} \bmod n. \tag{1}$$

2.3 Description

First, randomly select an integer g such that n divides the order of g. This can be done by checking whether

$$\gcd\left(\mathcal{L}(g^{\lambda} \bmod n^2), n\right) = 1. \tag{2}$$

The pair (n, g) is then published as the public key, whilst the pair (p, q) (or equivalently λ) forms the secret key. The cryptosystem is described on figure 1.

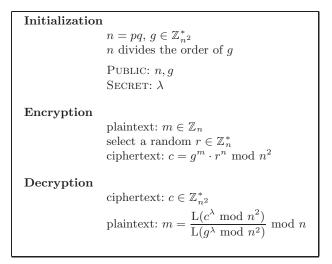


Fig. 1. Main Scheme

Decryption thus requires essentially one exponentiation modulo n^2 with exponent λ . As pointed out in [15], this encryption scheme is one-way if and only if the CRA holds.

2.4 The Subgroup Variant

In this variant (see figure 2), the idea consists in restricting the ciphertext space $\mathbb{Z}_{n^2}^*$ to the subgroup $\langle g \rangle$ of smaller order by taking advantage of the following extension of Equation 1. Assume that the order of g is $n\alpha$ for some $1 \leq \alpha \leq \lambda$. Then for any $w \in \langle g \rangle$,

$$\llbracket w \rrbracket_g = \frac{L(w^\alpha \mod n^2)}{L(g^\alpha \mod n^2)} \mod n.$$
 (3)

By carefully setting α to an integer of suitable length ℓ (typically 320 bits in practice), the decryption workload thus decreases to an exponentiation with an ℓ -bit exponent.

```
n = pq, \, \alpha | \lambda
h \in \mathbb{Z}_{n^2}^* \text{ of maximal order } n\lambda
g = h^{\lambda/\alpha} \mod n^2
\text{PUBLIC: } n, g
\text{SECRET: } \alpha
\text{Encryption}
\text{plaintext: } m \in \mathbb{Z}_n
\text{randomly select } r < 2^{\ell}
\text{ciphertext: } c = g^{m+nr} \mod n^2
\text{Decryption}
\text{ciphertext: } c \in \mathbb{Z}_{n^2}^*
\text{plaintext: } m = \frac{L(c^{\alpha} \mod n^2)}{L(g^{\alpha} \mod n^2)} \mod n
\text{if the computation was impossible, output "failure"}
```

Fig. 2. Subgroup Variant

Naturally, the problem of computing $\llbracket w \rrbracket_g$ for $w \in \langle g \rangle$ is (computationally) weaker than doing so for $w \in \mathbb{Z}_{n^2}^*$. For $\ell = \Omega\left(|n|^{\epsilon}\right)$ with $\epsilon > 0$, it is however considered to be intractable: this complexity hypothesis is known as the Partial Discrete Log (PDL) assumption. Moreover, inverting the hereabove encryption scheme was shown to be intractable if (and only if) the PDLA assumption holds.

2.5 Security Results

In a similar way, both cryptosystems were proven semantically secure against chosen-plaintext attacks (IND-CPA) under the additional complexity assumptions that the decisional versions of the Composite Residuosity and Partial Discrete Log problems are also intractable (see [15] for technical details). These intractability hypothesis are called Decision Composite Residuosity (D-CRA) and Decision Partial Discrete Log (D-PDLA) assumptions, respectively. Because of their obvious malleability, however, both cryptosystems do not resist chosenciphertext attacks.

Remark 1. Interestingly, adaptive attacks do not seem to allow a total break-down (secret key retrieval) of these encryption schemes, whilst they trivially do in Okamoto-Uchiyama's [14] as pointed out in their original paper.

In the next section, we show how to render these schemes secure against adaptive chosen-ciphertext attacks relatively to the D-CRA and the D-PDLA assumptions in the random oracle model.

3 Improved Cryptosystems

The notion of security against an adaptive chosen-ciphertext attack (IND-CCA2) was introduced by Rackoff and Simon [17] as the property that a cryptosystem must have to resist active adversaries. In this scenario, the adversary makes queries of her choice to a decryption oracle during two stages. After the first stage of queries (the "find" stage), the attacker chooses two messages and requests an encryption oracle to encrypt one of them, leaving to the oracle the (secret) choice of which one. The adversary then continues to query the decryption oracle (the "guess" stage) with ciphertexts of her choice. Finally, she tells her guess about the choice the encryption oracle made. If she correctly guesses with probability non-negligibly higher than one half, in polynomial time, then the encryption scheme is considered unsecure.

In this section, we propose a modification of the main scheme (c.f. section 2.3) which provides security in the sense of IND-CCA2 under the D-CR assumption in the random oracle model. At that point, note that OAEP [3] cannot be employed ad hoc for this purpose, unless using Paillier's trapdoor one-way permutation: this would lead to a practical but inefficient converted encryption scheme. Also, Fujisaki and Okamoto's conversion method [10] could theoretically be applied but would considerably decrease the decryption speed (as previously said, a re-encryption is necessary during the converted decryption process) and therefore significatively reduce the practical interest of the subgroup variant.

Instead of these approaches, we rather rely on modifications specifically adapted to the schemes. For this purpose, we use a *t*-bit random number and two hash functions $G, H : \{0,1\}^* \mapsto 0 \{1\}^{|n|}$ seen as random oracles. The encryption scheme is described on figure 3.

Theorem 1. Provided $t = \Omega(|n|^{\delta})$ for $\delta > 0$, Scheme 1 is semantically secure against adaptive chosen-ciphertext attacks under the Decision Composite Residuosity assumption in the random oracle model.

Proof. Let us consider an adversary $A = (A_1, A_2)$ against the semantic security of Scheme 1, where A_1 denotes the "find"-stage and A_2 the "guess"-stage. We then use this adversary to efficiently decide n-residuosity classes. Indistinguishability of encryptions will be due to the randomness of the oracle G, whereas adaptive attacks are covered thanks to the random oracle H.

Simulation of the Decryption Oracle Since we consider chosen-ciphertext attacks, the adversary has access, in both stages, to a decryption oracle \mathcal{D} that we have to simulate: when the attacker asks for a ciphertext c to be decrypted, the simulator checks in the query-answer history obtained from the random oracle H whether some entry leads to the ciphertext c and then returns m; otherwise, it

Initialization $n = pq, g \in \mathbb{Z}_{n^2}^*$ n divides the order of qPublic: n, gSecret: λ Encryption plaintext: $m < 2^{|n|-t-1}$ randomly select $r < 2^t$ $z = H(m, r)^n \mod n^2$ $M = m||r + G(z \mod n) \mod n$ ciphertext: $c = g^M z \mod n^2$ Decryption ciphertext: $c \in \mathbb{Z}_{n^2}^*$ $M = \frac{L(c^{\lambda} \bmod n^2)}{L(q^{\lambda} \bmod n^2)} \bmod n$ $z' = g^{-M}c \mod n$ $m'||r' = M - G(z') \bmod n$ if $H(m', r')^n = z' \mod n$ then the plaintext is m'otherwise output "failure"

Fig. 3. Encryption scheme secure against adaptive attacks (Scheme 1)

returns "failure". This provides a quasi-perfect simulation since the probability of producing a valid ciphertext without asking the query (m, r) to the random oracle H (whose answer a has to satisfy the test $a^n = z \mod n$) is upper-bounded by $1/\phi(n) \leq 2/n$, which is clearly negligible. This simulator can also be seen as a knowledge extractor, which provides the plaintext-awareness [3] of the scheme.

Semantic Security We will rely on the attacker A to design a distinguisher B for n-residuosity classes. Let (w, α) be a given instance of the D-CR problem: α is suspected to be the n-residuosity class of w.

Our distinguisher B first randomly chooses $u \in \mathbb{Z}_n$, $v \in \mathbb{Z}_n^*$ and $0 \le r < 2^t$ and then computes $z = w \cdot g^{-\alpha} v^n \mod n$ as well as $c = w \cdot g^u v^n \mod n^2$. It then runs A_1 and gets two messages m_0 and m_1 . B chooses a bit b and runs A_2 on the ciphertext c, supposed to be the ciphertext of m_b using the random r.

If during the game z is asked to the oracle G (which event will be denoted by AskG), one stops the game and B returns 1. If (m_0, r) or (m_1, r) are asked to the oracle H (which event will be denoted by AskH), one stops the game and B returns 0. In any other case, B returns 0 when A_2 ends.

One has to remark that no more than one event AskG or AskH is likely to happen since both events make B terminate the game. Furthermore, because of

the random choice of r, the probability of AskH is upper bounded by $q_H/2^t$ in any case, where q_H denotes the number of queries asked to H.

Since G and H are seen like random oracles, the attacker has no chance to correctly guess b, during a real attack (or in the case where $\alpha = \llbracket w \rrbracket_g$), if none of the events AskG or AskH occur, then $\mathsf{Adv}_A \leq \Pr[\mathsf{AskG} \vee \mathsf{AskH} | \llbracket w \rrbracket_g = \alpha]$.

On the other hand, if $\alpha \neq \llbracket w \rrbracket_g$, z is perfectly random (and furthermore independent of c), then AskG cannot occur with probability greater than $q_G/\phi(n)$, where q_G denotes the number of queries asked to G.

Therefore, the distinguisher B gets the following advantage in deciding the n-residuosity classes

$$\begin{split} \mathsf{Adv}_B &= \Pr[1|\, [\![w]\!]_g = \alpha] - \Pr[1|\, [\![w]\!]_g \neq \alpha] \\ &= \Pr[\mathsf{AskG}|\, [\![w]\!]_g = \alpha] - \Pr[\mathsf{AskG}|\, [\![w]\!]_g \neq \alpha] \\ &= \Pr[\mathsf{AskG} \vee \mathsf{AskH}|\, [\![w]\!]_g = \alpha] - \Pr[\mathsf{AskH}|\, [\![w]\!]_g = \alpha] - \Pr[\mathsf{AskG}|\, [\![w]\!]_g \neq \alpha] \\ &\geq \mathsf{Adv}_A - q_H/2^t - q_G/\phi(n) \geq \mathsf{Adv}_A - q_H/2^t - 2q_G/n. \end{split}$$

Reduction Cost To conclude, if there exists an active attacker A against semantic security, one can decide n-residuosity classes with an advantage greater than

$$\mathsf{Adv}_A \times \left(1 - \frac{2}{n}\right)^{q_D} - \frac{q_H}{2^t} - \frac{2q_G}{n} \geq \mathsf{Adv}_A - \frac{q_H}{2^t} - 2 \cdot \frac{q_G + q_D}{n},$$

where q_D , q_G and q_H denote the number of queries asked to the decryption oracle, G and H respectively.

However, the ability to decide n-residues may not be enough to break the semantic security. Furthermore, one can also prove that the converted scheme is still one-way relatively to the computational problem.

Theorem 2. Provided $t = \Omega(|n|^{\delta})$ for $\delta > 0$, Scheme 1 is one-way, even against adaptive chosen-ciphertext attacks, under the Composite Residuosity assumption in the random oracle model.

Proof. Let us consider an adversary A able to decrypt any ciphertext c with probability ε , within a time bound T. Since we consider chosen-ciphertext attacks, the adversary has access to a decryption oracle \mathcal{D} whose simulation works as described above. During an attack, two cases may appear

case 1: either the attacker tries to check the validity of the ciphertext, then she has to compute H(m,r) which gives us m, r and therefore the n-residuosity class of the ciphertext;

case 2: or she asks the query z to the oracle G (as previously noticed, she gets no information about the plaintext without such a query).

As already mentioned, the attacker cannot get any information about the plaintext if none of these cases applies. Then either the first case applies with probability greater than $\varepsilon/2$, and therefore the attacker can be used to compute n-residuosity classes with probability greater than $\varepsilon/2$ within a time bound T, or the second case applies with probability greater than $\varepsilon/2$. Let us consider the second case. For simplicity, we will restrict the study to the setting $3t \leq |n|$ since other parameter choices present no practical interest, and would make the proof be more intricate.

Let w be an element of $\mathbb{Z}_{n^2}^*$ of class α we want to compute. One randomly chooses $\alpha_0, \alpha_1 \in \mathbb{Z}_n, \beta_0, \beta_1 \in \mathbb{Z}_n^*$, and computes

$$w_0 = w \cdot g^{\alpha_0} \beta_0^n \mod n^2$$
 and $w_1 = w^{-2^t} \cdot g^{\alpha_1} \beta_1^n \mod n^2$.

The ciphertexts w_0 and w_1 are successively given to the attacker and all the answers ρ_i (randomly chosen in \mathbb{Z}_n by the simulation of G during the attack against w_0) as well as all the answers σ_i (given during the attack against w_1) are collected and stored. Because of the uniform distribution of w_0 and w_1 in $\mathbb{Z}_{n^2}^*$, the attacker A succeeds in correctly finding m_0 and m_1 , dropping in case 2, with probability $\varepsilon^2/4$. Then, there exist $0 \le r_0, r_1 < 2^t$ and indices i, j such that

$$\alpha + \alpha_0 = 2^t m_0 + r_0 + \rho_i \mod n$$
$$-2^t \cdot \alpha + \alpha_1 = 2^t m_1 + r_1 + \sigma_i \mod n.$$

By combination, one gets

$$2^{t}r_{0} + r_{1} = 2^{t}\alpha_{0} + \alpha_{1} - 2^{2t}m_{0} - 2^{t}m_{1} - 2^{t}\rho_{i} - \sigma_{j} \mod n.$$

Hence there exists at least one pair (i, j) such that

$$0 \le 2^t \alpha_0 + \alpha_1 - 2^{2t} m_0 - 2^t m_1 - 2^t \rho_i - \sigma_i \mod n < 2^{2t}.$$

One randomly chooses such a pair (i,j) which allows to compute r_0 and r_1 , and therefore α (with probability greater than $1/q_G^2$ if m_0 and m_1 are correct, since there are at most q_G^2 possible pairs (i,j), where q_G is the number of queries asked to G during a decryption). Consequently, our reduction recovers α with probability greater than $(\varepsilon/2q_G)^2$ within a time bound 2T.

However, this reduction can be heuristically shown much more efficient. Indeed, the probability of having one valid pair (i, j) for incorrect plaintexts (m_0, m_1) and the probability of having many pairs (i, j) for valid plaintexts m_0 , m_1 are both upper-bounded by $q_G^2/2^t$ (because of the randomness of the ρ_k and σ_k sequences, and perfect independence of $\alpha_0, \alpha_1, \beta_0, \beta_1$). We can therefore ignore them for a large enough security parameter t. Finally, one can recover α within an expected time bounded by $8T/\varepsilon$ (this is an optimal reduction).

At this point, we comment that Fujisaki and Okamoto's conversion technique [10] would have given an identical security level and a quite similar computational workload. The superiority of our approach resides in that

a) the one-wayness of our both converted schemes are equivalent to the CR and PDL problems, whereas Fujisaki-Okamoto would have inherently restricted one-wayness to the decision problems D-CR and D-PDL, b) the same proof as above will now apply almost unchanged on the subgroup variant, leading to a far better decryption efficiency (our validity test does not involve a complete re-encryption).

We now turn to show how to modify the subgroup variant (c.f. section 2.4) to meet NM-CCA2 security under the D-PDLA in the random oracle model. As before, we make use of two hash functions, $G, H : \{0,1\}^* \mapsto 0 \{1\}^{|n|}$ considered as random oracles. In what follows, we set α to an odd divisor $\alpha = 2a + 1$ of λ with bitsize ℓ and randomly pick an element h of maximal order $n\lambda$ in $\mathbb{Z}_{n^2}^*$. Recall that the modulus n is chosen such that p-1 and q-1 do not have common prime divisors other than 2. The encryption scheme is depicted on figure 4.

```
Initialization
                   n = pq, gcd(p - 1, q - 1) = 2, \alpha = 2a + 1|\lambda
                   h \in \mathbb{Z}_{n^2}^* of maximal order n\lambda
                   g = h^{\lambda/\alpha} \mod n^2
                   Public: n, q
                   Secret: \alpha
Encryption
                   plaintext: m < 2^{|n|-t-1}
                   randomly select r < 2^t
                   z = q^{nH(m,r)} \bmod n^2
                   M = m||r + G(z \bmod n) \bmod n
                   ciphertext: c = g^M z \mod n^2
Decryption
                   ciphertext: c \in \mathbb{Z}_{n^2}^*
                   M = \frac{L(c^{\alpha} \bmod n^2)}{L(q^{\alpha} \bmod n^2)} \bmod n
                   if the computation was impossible, output "failure"
                   z' = g^{-M}c \bmod n
                   m'||r' = M - G(z') \bmod n
                   if q^{nH(m',r')} = z' \mod n then the plaintext is m'
                   otherwise output "failure"
```

Fig. 4. Efficient variant secure against adaptive attacks (Scheme 2)

Theorem 3. Provided $t = \Omega(|n|^{\delta})$ and $\ell = \Omega(|n|^{\epsilon})$ for $\delta, \epsilon > 0$, Scheme 2 is semantically secure against adaptive chosen-ciphertext attacks under the D-PDLA in the random oracle model.

Proof. The proof is essentially the same, using $g^{H(m,r)}$ instead of H(m,r), but we also have to prove that one cannot decrypt a ciphertext which has not been

correctly computed (using the encryption scheme). Indeed, our simulation of the decryption oracle (the plaintext extractor) can only decrypt a valid ciphertext.

Let then c be an accepted ciphertext (i.e which has not caused a decryption failure). This means that $c^{\alpha} = 1 \mod n$ and thus $c^{\alpha} \in \mathcal{S}_n = \langle h^{\lambda} \rangle \subset \langle h \rangle$. We have assumed that $\alpha = 2a + 1$, $\gcd(p - 1, q - 1) = 2$ and h of maximal order $\lambda(n^2) = \phi(n^2)/2$. Then, for any $x \in \mathbb{Z}_{n^2}^*$, $x^2 \in \langle h \rangle$. Therefore,

$$c = c^{\alpha} \cdot c^{-2a} = c^{\alpha} \cdot (c^2)^{-a} \in \langle h \rangle$$
,

which implies the existence of an x such that $c=h^x \mod n^2$. Furthermore, $c^\alpha=h^{x\alpha}=1 \mod n$, and h is also of maximal order in \mathbb{Z}_n^* . Therefore there exists y such that $x=\beta y$, where $\lambda=\alpha\beta$. Thus, $c=h^{\beta y}=g^y \mod n^2$, and the n-residuosity class M obtained during the decryption process satisfies $M=y \mod n$ and $c=g^{M+kn} \mod n^2$. Hence $z=g^{kn} \mod n$.

Because the ciphertext was accepted, we have that $z = g^{nH(m',r')} \mod n$, and therefore $k = H(m',r') \mod \alpha$, because g is of order α in \mathbb{Z}_n^* , and so is g^n (n is relatively prime to ϕ). Finally, if we define m = m' and r = r', one gets c as ciphertext.

Theorem 4. Provided $t = \Omega(|n|^{\delta})$ and $\ell = \Omega(|n|^{\epsilon})$ for $\delta, \epsilon > 0$, Scheme 2 is one-way, even against adaptive chosen-ciphertext attacks, under the Partial Discrete Logarithm assumption in the random oracle model.

Remark 2. Because of the plaintext extractor presented in the proof of Theorem 1, both schemes are plaintext-aware [3].

4 Encryption Parameters

In practice, α should be typically set to a 320-bit divisor of λ such that $\alpha = \alpha_p \alpha_q$ where α_p divides p-1 but not q-1 and α_q divides q-1 but not p-1. This can be met using an appropriate key generation algorithm. Note that our converted schemes, like the original ones, allow Chinese remaindering for decryption. In the subgroup variant, interestingly, the form of α leads to two exponentiations modulo p^2 and q^2 with 160-bit exponents. This clearly shows one advantage of this encryption scheme in terms of decryption throughput.

Also, we fix t to 80, that is, we recommend that random numbers r have bitsize 80 or more in practical use.

5 Efficiency

This section gives tight estimates of our cryptosystems' running times for decryption compared to standard ones (OAEP, El Gamal). The elementary unit will be taken as the number of modular multiplications of bitsize |n| per kilobit of message input; it therefore depends on |n|. Typical modulus sizes are $|n| = 512, \dots, 2048$. We also assume that the execution time of a modular multiplication is quadratic in the operand size and that modular squares are computed

by the same routine. Chinese remaindering, as well as random number generation for probabilistic schemes, is considered to be negligible. The parameter t is set to 80 in our two cryptosystems. All secret grandeurs such as factors and exponents are assumed to contain about the same number of ones and zeroes in their binary representation.

We give purely indicative estimates which do not come from actual implementations. Pre-processing stages are not considered, but Chinese remaindering is taken into account whenever possible (hence for all schemes but El Gamal).

Schemes	Scheme 1	Scheme 2	OAEP	ElGamal
One-wayness	CR	PDL	RSA	DH
IND-CPA	D - CR	D - PDL	RSA	D-DH
NM-CCA2	D - CR	D - PDL	RSA	none
Plaintext size Ciphertext size	n - 80 $2 n $	n - 80 2 n	n - 320 $ n $	p 2 p

Decryption Work	load (Mult/Kbits)			
n , p = 512	2731	1707	1024	1536
n , p = 768	2572	1072	658	1536
n , p = 1024	2499	781	559	1536
n , p = 1536	2431	506	485	1536
n , p = 2048	2398	375	455	1536

6 Conclusion and Further Research

We proposed two new public-key cryptosystems provably semantically secure against adaptive chosen-ciphertext attacks *i.e.* secure in the sense of NM-CCA2. Computationally efficient for decryption, one of them could provide an alternative to OAEP. A typical research topic would be to ensure security against active adversaries relatively to the *computational* related problems CR and PDL. Another (independent) direction consists in improving their decryption throughputs by accelerating computations modulo p^2 , possibly using appropriate modular techniques such as [21].

References

 M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, 1998. 166

- M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of the First ACM CCCS*, pages 62–73. ACM Press, 1993. 166
- M. Bellare and P. Rogaway. Optimal Asymmetric Encryption How to Encrypt with RSA. In Eurocrypt '94, LNCS 950, pages 92–111. Springer-Verlag, 1995. 165, 166, 167, 171, 172, 176
- D. Bleichenbacher. A Chosen Ciphertext Attack against Protocols based on the RSA Encryption Standard PKCS #1. In Crypto '98, LNCS 1462, pages 1–12. Springer-Verlag, 1998. 165
- J. S. Coron, D. Naccache and Ju. Stern. A New Signature Forgery Strategy. In Crypto '99, Springer-Verlag, 1999. 165
- R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In Crypto '98, LNCS 1462, pages 13–25. Springer-Verlag, 1998. 167
- W. Diffie and M. E. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, volume IT–22, no. 6, pages 644–654, November 1976. 165
- 8. D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *Proc. of the 23rd STOC*. ACM Press, 1991. 166
- 9. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *IEEE Transactions on Information Theory*, volume IT—31, no. 4, pages 469–472, July 1985. 167
- E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In PKC '99, LNCS 1560, pages 53–68. Springer-Verlag, 1999. 165, 166, 167, 171, 174
- S. Goldwasser and S. Micali. Probabilistic Encryption. Journal of Computer and System Sciences, 28:270–299, 1984. 166
- D. Naccache and J. Stern. A New Cryptosystem based on Higher Residues. In Proc. of the 5th CCCS, pages 59–66. ACM press, 1998.
- M. Naor and M. Yung. Public-Key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In Proc. of the 22nd STOC, pages 427–437. ACM Press, 1990. 166
- T. Okamoto and S. Uchiyama. A New Public Key Cryptosystem as Secure as Factoring. In *Eurocrypt '98*, LNCS 1403, pages 308–318. Springer-Verlag, 1998. 167, 170
- P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithms Residues. In Eurocrypt '99, LNCS 1592, pages 223–238. Springer-Verlag, 1999. 167, 168, 169, 170
- D. Pointcheval. New Public Key Cryptosystems based on the Dependent-RSA Problems. In *Eurocrypt '99*, LNCS 1592, pages 239–254. Springer-Verlag, 1999.
- 17. C. Rackoff and D. R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto '91*, LNCS 576, pages 433–444. Springer-Verlag, 1992. 166, 171

- R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM, 21(2):120–126, February 1978. 165
- 19. RSA Data Security, Inc. Public Key Cryptography Standards PKCS. Available from http://www.rsa.com/rsalabs/pubs/PKCS/. 167
- V. Shoup and R. Gennaro. Securing Threshold Cryptosystems against Chosen Ciphertext Attack. In *Eurocrypt '98*, LNCS 1403, pages 1–16. Springer-Verlag, 1998. 167
- T. Takagi. Fast RSA-Type Cryptosystems Using N-adic Expansion. In Crypto '97, LNCS 1294, pages 372–384. Springer-Verlag, 1997. 177
- 22. Y. Tsiounis and M. Yung. On the Security of El Gamal based Encryption. In *PKC '98*, LNCS. Springer-Verlag, 1998. **167**

Adaptively-Secure Optimal-Resilience Proactive RSA

Yair Frankel*, Philip MacKenzie**, and Moti Yung* * *

Abstract. When attacking a distributed protocol, an adaptive adversary may determine its actions (e.g., which parties to corrupt), at any time, based on its entire view of the protocol including the entire communication history. In this paper we are concerned with proactive RSA protocols, i.e., robust distributed RSA protocols that rerandomize key shares at certain intervals to reduce the threat of long-term attacks. Here we design the first proactive RSA system that is secure against an adaptive adversaries. The system achieves "optimal-resilience" and "secure space scalability" (namely O(1) keys per user).

1 Introduction

Distributed public-key systems involve public/secret key pairs where the secret key is distributively held by some number of servers. As long as an adversary cannot corrupt a quorum of servers the system remains secure (as opposed to centralized cryptosystems in which the compromise of a single entity breaks the system). Distributed cryptography, including the design of practical distributed cryptosystems, has been an area of extensive research (see surveys [9,23,21]).

In this paper, we present the first proactive RSA system that is both optimally resilient and provably secure against an adaptive adversary. The system is also scalable in that it only requires a number of key shares that is linear in the number of servers. Let us review why such a proactive RSA system is considered one of the hardest amongst the "distributed cryptosystems" problems:

- 1. RSA is harder to "make distributed" than the discrete logarithm family due to mathematical constraints. The group of RSA exponents (which contains the RSA secret key) has unknown order (as opposed to discrete-log based protocols), which makes distributed manipulations with the partial keys: polynomial interpolation and key re-randomization (typical in "proactive systems" [28]) hard.
- 2. From an adversarial perspective, the "proactive" system is the most difficult since it must cope with a "mobile" adversary (whereas a regular "robust threshold system" copes only with static adversary)..

^{*} CertCo, yfrankel@cs.columbia.edu

^{**} Information Sciences Research Center, Bell Laboratories, Murray Hill, NJ 07974, philmac@research.bell-labs.com

^{* * *} CertCo, moti@cs.columbia.edu

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 180–195, 1999. © Springer-Verlag Berlin Heidelberg 1999

- 3. For l denoting the number of servers and t the number of misbehaving servers, an "optimally resilient" protocol is the weakest possible constraint, of $l \geq 2t + 1$, (matching the trivial lower bound).
- 4. The size of keying information per server is an important parameter in determining scalability of a system. For "scalability of secure space," we require each server to store O(1) key shares, which is optimal.
- 5. The security proof for distributed protocols is usually based on simulating the adversary's view of the protocol. From the point of view of "proving security" against an adaptive adversary using simulatability is the most difficult one because the view of an adaptive adversary changes dynamically based on all information it receives.

Formally, what we show is:

Theorem 1. There exists an adaptively-secure optimal-resilient space-scalable proactive RSA system.

2 Background

The "adaptive adversary" challenge:

Let us elaborate more on the problem facing the designers of adaptively-secure distributed protocols. (Very few such systems have been designed e.g., [1,2]). The major difficulty in proving the security of protocols against adaptive adversaries is being able to efficiently simulate (without actually knowing the secret keys) the view of an adversary which may corrupt parties dynamically, depending on its internal "unknown strategy." The adversary's corruption strategy may be based on values of public ciphertexts, other public cryptographic values in the protocol, and the internal states of previously corrupted parties. Let a user's share of the private key be committed by any public value which is based on the share. The commitment may be an encryption of the share, or the computation to generate the signature with the quorum. Suppose the adversary decides to corrupt next the party whose identity matches a one-way hash function applied to the entire communication history of the protocol so far. Now we must simulate the corruption which allows obtaining a share of the key which is consistent with the commitments.

In distributed public-key systems, the problem of adaptive security is exacerbated by the fact that there is generally "public function and related publicly-committed robustness information" available to anyone, which as discussed above, needs to be consistent with internal states of parties that get corrupted. This is the main cause of difficulties in the proof of security. With proactive systems the update has to be done correctly and be connected to the past, which is another source of difficulty when facing an adaptive adversary.

History of proactive RSA:

The work of [24] developed tools to allow proactive security [28] in discrete-log based systems. They also defined the notion of a proactive public-key system. Tools to allow proactive security in RSA-based systems were given in [16,15,30].

(Previous work on threshold and robust threshold RSA systems is given in [10,8,17,22].) None of these tools have been shown to be secure against an adaptive adversary. Recently the notion of security against adaptive adversaries in threshold public-key systems was dealt with (in systems less constrained than ours) [19,3].

Our Contributions and Techniques:

We base our system on the one in [15], since it is optimally resilient, and, as opposed to [16,30], is secure-space scalable (requires only a linear number of RSA key shares). The system in [30] needs $O(l^3)$ RSA key shares. The system in [16] is suitable for small number of participants, but it is not optimally resilient, and may employ an even larger number of keys as the system grows in scale. Our system differs from the one in [15] in that we construct and employ a new set of techniques that make the system secure against an adaptive adversary.

As in previous proactive RSA systems, we perform secret sharing operations over the integers (since for security the order of the group containing the secret key cannot be revealed to the servers). A difficulty with integer operations is maintaining bounded share sizes even after many proactive update (rerandomization) operations. We do this without requiring any zero-knowledge proofs in our update protocol. In fact, we only use zero-knowledge proofs in our function application (signature) protocol, and if one wishes to optimize the system for the corruption-free case, these proofs would only need to be performed if the computed signature is found to be invalid (which is the typical "optimistic" approach to fault-tolerance advocated e.g. in [16]).

Since we must maintain security against an adaptive adversary, we are not able to use Feldman's verifiable secret sharing VSS [14], which is inherently insecure against an adaptive adversary. (Previous protocols, such as [15,30] are based on Feldman VSS, and [30] uses it in particular to avoid the use of zeroknowledge proofs and thus increase some of the efficiency.) Instead, we use a form of Pedersen's VSS [29]. One can think of the commitments in this VSS as "detached commitments". These commitments are used to ensure correct behavior of servers, yet have no "hard attachment" to the rest of the system, even the secret key itself! We show how to work with these detached commitments, e.g., using "function representation transformations" like "poly-to-sum" and "sum-to-poly" (which are basic tools which maintain space-scalability and which we build based on [15]). We also show how to maintain robustness by constructing simulatable "soft attachments" from these detached commitments into the operations of the rest of the system. The soft attachments are constructed using efficient zero-knowledge proofs-of-knowledge. For proactive maintenance we develop update techniques that carefully assure correctness of update, as well as non-growth of the key size as we perform our computations over the integers.

The basic proof technique for security against an adaptive adversary is based on the notion of a "faking server" which is chosen when commitments related to the secret key must be generated. (Fortunately, this is only during a function application phase, where additive sharings are used.) The simulator exploits the "actions" of this server to assure that the view is simulatable while not

knowing the secret key. This server is chosen at random and its public actions are indistinguishable from an honest server to the adversary. We have to backtrack the simulation **only if** the adversary corrupts this special server. Since there is only one faking server, and since regardless of its corruption strategy, the adversary has a polynomial chance (at least 1/(t+1)) of not corrupting this one server, we will be able to complete the simulation in expected polynomial time. We have to assure that this "faking server" technique works in the proactive setting, where the adversary is not just adaptive, but also mobile. Our protocol and simulation techniques also maintain "optimal resilience."

3 Model and Definitions

Participants and Communication: We use the standard model for proactive systems [28]. The system consists of l servers $S = \{S_1, \ldots, S_l\}$. A server is corrupted if it is controlled by the adversary. When corrupted, we assume "for security" that the adversary sees all the information currently on that server. On the other hand, the system should not "open" secrets of unavailable servers (effectively reducing the needed threshold). Namely, we separate availability faults from security faults (and do not cause potential security exposures due to unavailability). Our communication model is similar to [25]. All participants communicate via an authenticated bulletin board in a synchronized manner. We assume that the adversary cannot jam communication.

Time periods: To deal with a mobile adversary, we assume a common global clock (e.g., a day, a week, etc.) that divides time into two types of *time periods* repeated in sequence: an **update period** (odd times) and an **operational period** (even times). During an operational period, the servers can perform functions using the current secret key shares. During the update period the servers engage in an interactive *update protocol* which upon completion the servers hold new shares to be used during the following operational period¹.

System Management: We assume that a server that is determined to be corrupted by a majority of active servers can be *refreshed* (i.e., erased and rebooted, or perhaps replaced by a new server with a blank memory) by some underlying system management. This is a necessary assumption for dealing with corrupted servers in any proactive system.

The adversary: The adversary is t-restricted; namely it can, during each period, corrupt at most t servers, under the assumption that if a server is corrupt during an update period it is corrupt in the prior and subsequent operational periods. A mobile adversary moves around as servers become corrupted or uncorrupted. The actions of an adversary at any time may include submitting messages to

¹ Technical note: we consider a server that is corrupted during an update phase as being corrupted during both its adjacent periods. This is because the adversary could learn the shares used in both the adjacent operational periods.

the system to be signed, corrupting servers, and broadcasting arbitrary information on the communication channel. The adversary is **adaptive**; namely it is allowed to base its actions not only on previous function outputs, but on all the information that it has previously obtained.

Distributed Public-Key Systems: Here we formally define our notions of security and robustness for proactive public-key systems. For simplicity, we will assume that the function application operation computes signatures.

Definition 1. (Robustness of a Proactive System) A(t,l)-proactive publickey system S is robust if for any polynomial-time t-restricted adaptive mobile adversary A, with all but negligible probability, after polynomially-many update protocols for each input m which is submitted to the signing protocol during an operational period, the resulting signature s is valid.

Definition 2. (Security of a Proactive System) A(t,l)-proactive publickey system S is secure if for any polynomial-time t-restricted adaptive mobile adversary A, after polynomially-many signing protocols performed during operational periods on given values, and polynomially-many update protocols, given a new value m and the view of A, the probability of being able to produce in polynomial time a signature s on m that is valid is negligible.

Remark: The choice of the inputs to the signing protocol prior to the challenge m defines the $tampering\ power$ of the adversary (i.e., "known message," "chosen message", "random message" attacks). The choice depends on the implementation within which the distributed system is embedded. In this work, we assume that the (centralized) cryptographic function is secure with respect to the tampering power of the adversary. We note that the provably secure signature and encryption schemes typically activate the cryptographic function on random values (decoupled from the message choice of the adversary).

3.1 Range Notation

We will use the following notation to define ranges:

- 1. A range is defined as [a, b], denoting all integers between a and b inclusive. (Similarly, we could define (a, b), (a, b], and [a, b).)
- 2. We can multiply a range by a positive scalar: $x \cdot [a, b]$ denotes the range [xa, xb].
- 3. We can multiply a range by a negative scalar: $-x \cdot [a, b]$ denotes the range [-xb, -xa].
- 4. We can add two ranges: [a, b] + [c, d] denotes the range [a + c, b + d].
- 5. For a range that covers the point zero, we can multiply that range by a positive or negative scalar: $\pm x \cdot [a, b]$, where $a \leq 0 \leq b$, denotes the range [-xc, xc], where $c = \max(|a|, |b|)$.

4 Basics for Our System

Let k be the security parameter. Let key generator GE define a family of RSA functions to be $(e,d,N) \leftarrow GE(1^k)$ such that N is a composite number N=P*Q where P,Q are prime numbers of k/2 bits each. The exponent e and modulus N are made public while $d \equiv e^{-1} \mod \lambda(N)$ is kept private. The **RSA encryption function** is public, defined for each message $M \in Z_N$ as: $C = C(M) \equiv M^e \mod N$. The **RSA decryption function** (also called signature function) is the inverse: $M = C^d \mod N$. It can be performed by the owner of the private key d. Formally the RSA Assumption is stated as follows.

RSA Assumption Let k be the security parameter. Let key generator GE define a family of RSA functions (i.e., $(e,d,N) \leftarrow GE(1^k)$ is an RSA instance with security parameter k). For any probabilistic polynomial-time algorithm A, $\Pr[u^e \equiv w \mod N : (e,d,N) \leftarrow GE(1^k); w \in_R \{0,1\}^k; u \leftarrow A(1^k, w, e, N)]$ is negligible.

Recall that the RSA assumption implies the intractability of factoring products of two large primes.

Next we describe variants of Shamir secret sharing and Pedersen VSS that we use. They differ in that operations on the shares are performed over the integers, instead of in a modular subgroup of integers.

(t,l)-secret sharing over the integers [15]: This primitive is based on Shamir secret sharing [32]. Let L=l! and let β,K be positive integers. For sharing a secret $s\in[0,K]$, a random polynomial $a(x)=\sum_{j=0}^t a_jx^j$ is chosen such that $a_0=L^2s$, and each other $a_j\in_R\{0,L,2L,\ldots,\beta L^3K\}$. Each shareholder $i\in\{1,\ldots,l\}$ receives a secret share $s_i=a(i)$, and verifies that $(1)\ 0\leq s_i\leq\beta L^3Kl^t(t+1)$, and $(2)\ L$ divides s_i . Any set of shareholders of cardinality t+1 can compute s using Lagrange interpolation, i.e. $s=a(0)=\sum_{i\in\Lambda}a(i)z_{i,\Lambda}$, where $z_{i,\Lambda}=\prod_{j\in\Lambda\setminus\{i\}}(i-j)^{-1}(0-j)$.

(t,l)-Unconditionally-Secure VSS over the Integers (INT-(t,l)-US-VSS): This primitive is based on Pedersen Unconditionally-Secure (t,l)-VSS [29], and is slightly different than the version in [20]. Let N be an RSA modulus and let g and h be maximal order elements whose discrete log modulo N with respect to each other is unknown. The protocol begins with two (t,l)-secret sharings over the integers with $\beta = N$, the first sharing secret $s \in [0,K]$, and the second sharing $s' \in [0,N^2K]$. Let $a(x) = \sum_{j=0}^t a_j x^j$ be the random polynomial used in

 $^{^{2}}$ $\lambda(N) = \operatorname{lcm}(P-1, Q-1)$ is the smallest integer such that $x^{\lambda(N)} \equiv 1 \mod N$ for any $x \in \mathbb{Z}_{N}^{*}$. We use $\lambda(N)$, rather than more traditionally $\phi(N)$, because it explicitly describes an element of maximal order in \mathbb{Z}_{N}^{*} .

³ We note that L^2s is actually the secret component of the secret key, which when added to a public leftover component (in $[0, L^2 - 1]$), forms the RSA secret key.

⁴ These tests only verify the shares are of the correct form, not that they are correct polynomial shares.

sharing s and let $a'(x) = \sum_{j=0}^t a'_j x^j$ be the random polynomial used in sharing s'. For all i, S_i receives shares $s_i = a(i)$ and $s'_i = a'(i)$. (We refer to the pair (a(i), a'(i)) as dual-share i.) Also, the verification shares $\{\alpha_j (= g^{a_j} h^{a'_j})\}_{0 \le j \le t}$, are published. Call check share $A_i = \prod_{j=0}^t \alpha_j^{i^j}$. S_i can verify the correctness of his shares by checking that $A_i \stackrel{?}{=} g^{s_i} h^{s'_i}$. Say s and s' are the shares computed using Lagrange interpolation from a set of t+1 shares that passed the verification step. If the dealer can reveal different secrets \hat{s} and \hat{s}' that also correspond to the zero coefficient verification share, then the dealer can compute an α and β such that $g^{\alpha} \equiv h^{\beta}$, which implies factoring (and thus breaking the RSA assumption).

Looking ahead, we will need to simulate an INT-(t, l)-US-VSS. We can do this by constructing a random polynomial over an appropriate simulated secret (e.g., a random secret, or a secret obtained as a result of a previously simulated protocol) in the zero coefficient, and a random companion polynomial with a totally random zero coefficient.

5 Techniques

In order to "detach" ciphertexts from their cleartext values we employ semantically-secure non-committing encryption [1]. In fact, our (full) security proofs first assume perfectly secret channels and then add the above (a step we omit here).

A more involved issue concerns the public commitments. The collection of techniques needed to underly distributed public-key systems include: distributed representation methods (polynomial sharing, sum (additive) sharing), representation transformers which move between different ways to represent a function (poly-to-sum, sum-to-poly), as well as a set of "elementary" distributed operations (add, multiply, invert). For example, the "poly-to-sum" protocol is executed by t+1 servers at a time, and transforms a (t,l)-secure polynomial-based sharing to an additive sharing with t+1 shares. We need to have such techniques (motivated by [20,15]) which are secure and robust against adaptive adversaries. We will rely on new zero-knowledge proof techniques (see Appendix A), as well as on shared representation of secrets as explained in Section 4. The notation "2poly" refers to a polynomial and its companion polynomial shared with INT-(t, l)-US-VSS (which is "unconditionally secure VSS"). The notation "2sum" refers to two additive sharings, with check shares that contain both additive shares of a server (similar to the check shares in INT-(t, l)-US-VSS). In describing the protocols, unless otherwise noted we will assume multiplication is performed mod N and addition (of exponents) is performed over the integers (i.e., not "mod" anything).

We define the following ranges, using the notation described in Section 3.1.

- 1. Let $[RANGE_1] = [0, (t+1)L^3N^3l^t]$ and $[RANGE_1] = [0, (t+1)L^3(t+1)N^7l^t]$.
- 2. Let $[RANGE_2] = \pm (t+1)L \cdot [RANGE_1]$ and $[RANGE_2'] = \pm (t+1)L \cdot [RANGE_1']$.
- 3. Let $[RANGE_3] = \pm (t+1) \cdot [RANGE_2] + [0, L^2N] + (t+1) \cdot [0, L^2]$ and $[RANGE_3'] = \pm (t+1) \cdot [RANGE_2'] + [0, L^2N^3] + (t+1) \cdot [0, L^2]$.

 $^{^5}$ We implicitly assume all verification operations are performed in $Z_N^\ast.$

- 4. Let $[RANGE_4] = (t+1) \cdot [RANGE_2] + (t+1) \cdot [0, L^2] + [RANGE_3]$ and $[RANGE'_4] = (t+1) \cdot [RANGE'_2] + (t+1) \cdot [0, L^2] + [RANGE'_3]$.
- 5. Let $[Range_5] = L \cdot [Range_4] (t+1) \cdot [0, L^2N^2/(t+1)]$ and $[Range_5'] = L \cdot [Range_4'] (t+1) \cdot [0, L^2N^5]$.

5.1 2poly-to-2sum

The goal of 2poly-to-2sum (Figure 1) is to transform t-degree polynomials a() and a'() used in INT-(t, l)-US-VSS into t+1 additive shares for each secret a(0) and a'(0), with corresponding check shares. The idea is to perform interpolation. We note that in Step 2 each s_i and s'_i is a multiple of L, so S_i can actually compute b_i and b'_i over the integers.

- 1. **Initial configuration:** INT-(t,l)-US-VSS (parameters: (N,g,h)) with t-degree polynomials a() and a'(), and a set Λ of t+1 server indices. For all $i \in \Lambda$, recall S_i holds shares s_i and s'_i with corresponding check share $A_i = g^{s_i} h^{s'_i}$.
- 2. For all $i \in \Lambda$, S_i computes the additive shares $b_i = s_i z_{i,\Lambda}$ and $b'_i = s'_i z_{i,\Lambda}$ and publishes $B_i = g^{b_i} h^{b'_i} = A_i^{z_{i,\Lambda}}$.
- 3. All servers verify B_i for all $i \in \Lambda$ using $(A_i)^{V_{i,\Lambda}} \equiv (B_i)^{V'_{i,\Lambda}}$ where $V_{i,\Lambda} = \prod_{j \in \Lambda \setminus \{i\}} (0-j)$ and $V'_{i,\Lambda} = \prod_{j \in \Lambda \setminus \{i\}} (i-j)$. If the verification for a given B_i fails, each server broadcasts a (Bad,i) message and quits the protocol.

Fig. 1. 2poly-to-2sum

Note the following ranges:

- For $i \in \Lambda$, if S_i is good then $s_i \in [RANGE_4]$ and $s_i' \in [RANGE_4']$. (This will be shown later.)

5.2 2sum-to-2sum

The goal of 2sum-to-2sum is to randomize additive dual-shares (most likely obtained from a 2poly-to-2sum) and update the corresponding check shares. The scheme is in Figure 2.

Note the following ranges:

- For $i \in \Lambda$, if S_i is good then $d_i \in [0, L^2N^2]$ and $d'_i \in [0, (t+1)L^2N^4]$.
- We will show that d_* ∈ [RANGE₃] and d'_* ∈ [RANGE'₃] (unless RSA is broken).

⁶ In [15], poly-to-sum also performed a rerandomization of the additive shares. We split that into a separate protocol for efficiency, since sometimes 2poly-to-2sum is used without a rerandomization of additive shares.

- 1. Initial configuration: There is a set Λ of t+1 server indices. For all $i \in \Lambda$, S_i holds additive dual-share (b_i, b'_i) , with corresponding check share $B_i = g^{b_i} h^{b'_i}$.
- 2. For all $i \in \Lambda$, S_i chooses $r_{i,j} \in \mathbb{R}$ $Z_{L^2N^2/(t+1)}$ and $r'_{i,j} \in \mathbb{R}$ $Z_{L^2N^4}$, for $j \in \Lambda$.
- 3. For all $i \in \Lambda$, S_i publishes $r_{i,*} = b_i \sum_{j \in \Lambda} r_{i,j}$ and $r'_{i,*} = b'_i \sum_{j \in \Lambda} r'_{i,j}$.
- 4. For all $i \in A$, S_i privately transmits $r_{i,j}$ and $r'_{i,j}$ to all S_j for $j \in A \setminus \{i\}$.
- 5. For all $i \in \Lambda$, S_i publishes $R_{i,j} = g^{r_{i,j}} h^{r'_{i,j}}$ for $j \in \Lambda \setminus \{i\}$.
- 6. All servers can compute $R_{i,i} = B_i/g^{r_{i,*}}h^{r'_{i,*}}\prod_{j\in\Lambda\setminus\{i\}}R_{i,j}$ for all $i\in\Lambda$.
- 7. For all $j \in A$, S_j verifies that each $r_{i,j} \in Z_{L^2N^2/(t+1)}$, each $r'_{i,j} \in Z_{L^2N^4}$, each $r_{i,*} \in [\text{RANGE}_5]$, each $r'_{i,*} \in [\text{RANGE}_5']$, and that $R_{i,j} \equiv g^{r_{i,j}} h^{r'_{i,j}}$. If the verification fails, S_j broadcasts an (Accuse, i,j) message, to which S_i responds by broadcasting $r_{i,j}$ and $r'_{i,j}$. If S_i does not respond, $r_{i,j}$ or $r'_{i,j}$ is not in the correct range, or $R_{i,j} \not\equiv g^{r_{i,j}} h^{r'_{i,j}}$ (which all servers can now test), then each server broadcasts a (Bad, i) message and quits the protocol.
- 8. For all $j \in \Lambda$, S_j computes $d_j = \sum_{i \in \Lambda} r_{i,j}$, $d'_j = \sum_{i \in \Lambda} r'_{i,j}$, and $D_j = \prod_{i \in \Lambda} R_{i,j}$.
- 9. All servers compute leftover shares $d_* = \sum_{i \in \Lambda} r_{i,*}$ and $d'_* = \sum_{i \in \Lambda} r'_{i,*}$

Fig. 2. 2sum-to-2sum

5.3 2sum-to-1sum

2sum-to-1sum (Figure 3) is employed in computing partial signatures using the first parts of the additive dual-shares (obtained from 2sum-to-2sum) as the exponents, and in proving the partial signatures correct. These proofs form the "soft attachments" from the information-theoretically secure check shares to the computationally secure check shares that must correspond to the actual secret.

- 1. Initial configuration: Parameters (N, e, g, h). There is a set Λ of t+1 server indices. For all $i \in \Lambda$, S_i holds additive dual-share (d_i, d'_i) , with corresponding check share $D_i = g^{d_i} h^{d'_i}$. Also, all servers S_i with $i \in \Lambda$ have performed a ZK-proof-setup protocol ZKSETUP-RSA(N, e, g) with all other servers.
- 2. For all $i \in \Lambda$, S_i broadcasts $E_i = m^{d_i}$, where m is the message to be signed, or more generally, the value to which the cryptographic function is being applied.
- 3. For all $i \in \Lambda$, S_i performs a ZK-proof of knowledge ZKPROOF-DL-REP $(N, e, m, g, h, E_i, D_i)$ with all other servers. Recall that this is performed over a broadcast channel so all servers can check if the ZK-proof was performed correctly.
- 4. If a server detects that for some $i \in \Lambda$, S_i fails to perform the ZK-proof correctly, that server broadcasts a message (Bad,i) and quits the protocol.

Fig. 3. 2sum-to-1sum

5.4 2sum-to-2poly

The goal of 2sum-to-2poly is to transform t+1 additive dual-shares with their corresponding check shares (obtained from 2sum-to-2sum) into polynomial sharings of the same secrets. The idea is for each shareholder to perform an INT-(t,l)-US-VSS with its shares as the secrets, and then to sum the resulting polynomials. The protocol is given in Figure 4

- 1. **Initial configuration:** Parameters (N, e, g, h). There is a set Λ of server indices. For all $i \in \Lambda$, S_i holds additive dual-share (d_i, d'_i) , with corresponding check share $D_i = g^{d_i}h^{d'_i}$, and all servers know leftover shares d_* and d'_* .
- 2. For $i \in \Lambda$, S_i broadcasts $r_i = d_i \mod L^2$ and $r'_i = d'_i \mod L^2$.
- 3. For $i \in \Lambda$, S_i sets $e_i = d_i r_i$, $e'_i = d'_i r'_i$, and $E_i = g^{e_i} h^{e'_i}$.
- 4. Note that all e_i and e'_i are multiples of L^2 . These are then shared using INT-(t,l)-US-VSS (over the appropriate ranges, and without any additional L^2 factor), say with polynomials $v_i()$ and $v'_i()$.
- 5. For all $j \in A$, S_j verifies that each $r_i, r'_i \in Z_{L^2}$, and verifies its shares of each INT-(t, l)-US-VSS. (For ranges, S_j must verify that each $v_i(j) \in [RANGE_1]$ and each $v'_i(j) \in [RANGE'_1]$.) If a verification fails for the INT-(t, l)-US-VSS from S_i , S_j broadcasts an (Accuse, i, j) message, to which S_i responds by broadcasting $v_i(j)$ and $v'_i(j)$. If S_i does not respond, $v_i(j)$ or $v'_i(j)$ is not in the correct range, or the verification shares do not match for $v_i(j)$ and $v'_i(j)$ (which all servers can now test), then each server broadcasts a (Bad, i) message and quits the protocol.
- 6. For all j, S_j computes the sums $v(j) = d_* + \sum_{i \in \Lambda} r_i + \sum_{i \in \Lambda} v_i(j)$ and $v'(j) = d'_* + \sum_{i \in \Lambda} r'_i + \sum_{i \in \Lambda} v'_i(j)$. The verification shares for v() and v'() can be computed from the verification shares for $v_i()$ and $v'_i()$, for $i \in \Lambda$.

Fig. 4. 2sum-to-2poly

Note the following ranges:

- For $i \in \Lambda$, if S_i is good then $e_i \in [0, L^2N^2]$ and $e'_i \in [0, (t+1)L^2N^4]$. Also, there is a range of size L^2N^4 for which an additive part of e'_i was randomly chosen by S_i itself, and this provides the simulatability of $v_i()$.
- Assuming (t+1) good servers check the ranges of their shares of $v_i()$ and $v_i'()$, then it can be deduced that $v_i(0) \in [RANGE_2]$ and $v_i'(0) \in [RANGE_2']$.
- Since $d_i = v_i(0)$ and $d_i' = v_i'(0)$ (unless RSA is broken, as we will prove), and since $d_* + \sum_{i \in \Lambda} d_i = L^2 s$ and $d_*' + \sum_{i \in \Lambda} d_i' = L^2 s'$ (again, unless RSA is broken), we have that $d_* \in [\text{RANGE}_3]$ and $d_*' \in [\text{RANGE}_3']$.
- Using the facts above, we conclude that for $j \in \Lambda$, if S_j is good then $v(j) \in [RANGE_4]$ and $v'(j) \in [RANGE'_4]$, i.e., for the next operational round, $s_j \in [RANGE_4]$ and $s'_j \in [RANGE'_4]$, which is what we stated in the 2poly-to-2sum protocol.

6 RSA Protocols

We now present protocols for threshold function application and proactive maintenance. Their security and robustness proofs are available from the authors [18].

6.1 Threshold Function Application

The protocol is given in Figure 5

- 1. Initial configuration: INT-(t, l)-US-VSS (parameters: (N, e, g, h)) with t-degree polynomials a() and a'(). Also, each server maintains a list $\mathcal G$ of server indices for servers that have not misbehaved (i.e., they are considered good). A message m needs to be signed.
- 2. A set $\Lambda \subseteq \mathcal{G}$ with $|\Lambda| = t + 1$ is chosen in some public way.
- 3. 2poly-to-2sum is run. If there are misbehaving servers, their indices are removed from \mathcal{G} and the protocol loops to Step 2.
- 4. 2sum-to-2sum is run. If there are misbehaving servers, their indices are removed from \mathcal{G} and the protocol loops to Step 2.
- 5. 2sum-to-1sum is run. If there are misbehaving servers, their indices are removed from \mathcal{G} and the protocol loops to Step 2. If there is no misbehavior, the signature on m can be computed from the partial signatures generated in this step, along with the leftover shares.
- 6. All values created during the signing protocol for m are erased.

Fig. 5. Function Application Protocol

6.2 Proactive Maintenance

For proactive maintenance, we perform an update by running 2poly-to-2sum on the secret polynomials, followed by 2sum-to-2sum and 2sum-to-2poly. After 2sum-to-2poly, each server erases all previous share information, leaving just the new polynomial shares and verification shares. If there is misbehavior by any server, the procedure is restarted with new participants (here restarts do not introduce statistical biases and do not reduce the protocol's security). The protocol is given in Figure 6, where the full Proactive RSA protocol is in Figure 7.

References

- D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In Advances in Cryptology—EUROCRYPT 92, volume 658 of Lecture Notes in Computer Science, pages 307–323. Springer-Verlag, 24–28 May 1992. 181, 186
- 2. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In STOC'96 [33], pages 639–648. 181
- 3. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive security of threshold systems. to appear in Crypto'99, 1999. 182
- 4. R. Cramer. Modular Design of Secure yet Practical Cryptographic Protocols. PhD thesis, University of Amsterdam, 1995. 193

- 1. **Initial configuration:** INT-(t, l)-US-VSS (parameters: (N, e, g, h)) with t-degree polynomials a() and a'()
- 2. Each server maintains a list \mathcal{G} of server indices for servers that have not misbehaved (i.e., they are considered good).
- 3. Each (ordered) pair of servers (S_i, S_j) performs ZKSETUP-RSA_{Si,Sj}(N, e, g, h) (using new commitment values). This setup will be used for all proofs performed during the following operational period.
- 4. A set $\Lambda \subseteq \mathcal{G}$ with $|\Lambda| = t + 1$ is chosen in some public way.
- 5. 2poly-to-2sum is run. If there are misbehaving servers, their indices are removed from \mathcal{G} and the protocol loops to Step 4.
- 6. 2sum-to-2sum is run. If there are misbehaving servers, their indices are removed from \mathcal{G} and the protocol loops to Step 4.
- 7. 2sum-to-2poly is run. If there are misbehaving servers, their indices are removed from \mathcal{G} and the protocol loops to Step 4.
- 8. All previous share information is erased.

Fig. 6. Proactive Maintenance (Key Update) Protocol

- R. Cramer, I. Damgård, and P. MacKenzie. Zk for free: the case of proofs of knowledge. manuscript, 1999.
- Advances in Cryptology—CRYPTO '89, volume 435 of Lecture Notes in Computer Science. Springer-Verlag, 1990, 20–24 Aug. 1989. 191, 193
- Advances in Cryptology—CRYPTO '91, volume 576 of Lecture Notes in Computer Science. Springer-Verlag, 1992, 11–15 Aug. 1991. 191, 193
- A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely (extended summary). In Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing, pages 522–533, Montréal, Québec, Canada, 23–25 May 1994.
- 9. Y. Desmedt. Threshold cryptosystems. In J. Seberry and Y. Zheng, editors, Advances in Cryptology—AUSCRYPT '92, volume 718 of Lecture Notes in Computer Science, pages 3–14, Gold Coast, Queensland, Australia, 13–16 Dec. 1992. Springer-Verlag. 180
- Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures (extended abstract). In CRYPTO'91 [7], pages 457–469. 182
- C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In STOC'98 [34], pages 409–428. 193
- 12. C. Dwork and A. Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In Krawczyk [27], pages 442–457. 193
- 13. U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In CRYPTO'89 [6], pages 526-545. 193
- P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In 28th Annual Symposium on Foundations of Computer Science, pages 427–437, Los Angeles, California, 12–14 Oct. 1987. IEEE. 182
- Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Optimal-resilience proactive public-key cryptosystems. In 38th Annual Symposium on Foundations of Computer Science, pages 384–393, Miami Beach, Florida, 20–22 Oct. 1997. IEEE. 181, 182, 185, 186, 187

- 1. The dealer generates an RSA public/private key (N, e, d), and computes public value x_{pub} and secret value $x \in [0, N]$ such that $d \equiv x_{pub} + L^2x \mod \phi(N)$, as in [16]. Then the dealer chooses generators $g, h \in_R Z_N^*$, $x' \in_R Z_{N^3}$, and an INT-(t, l)-US-VSS on secrets x, x' with parameters (N, g, h).
- 2. Each (ordered) pair of servers (S_i, S_j) performs $ZKSETUP-RSA_{S_i,S_j}(N, e, g, h)$.
- 3. Each server maintains a list \mathcal{G} of server indices for servers that have not misbehaved (i.e., they are considered good). It also maintains public parameters: $(N, e, g, h, x_{pub}, l, t)$ and the verification shares of the INT-(t, l)-US-VSS polynomials a() and a'() which have a(0) = x and a'(0) = x'.
- 4. When a message m needs to be signed, the servers agree on the public parameters, then the Function Application protocol is run.
- 5. When an update is scheduled to occur, the servers agree on the public parameters, then the Proactive Maintenance protocol is run.
- ^a Recall that x_{pub} is computed using only the public values N, e, L.

Fig. 7. Proactive Threshold Protocol

- Y. Frankel, P. Gemmell, P. D. MacKenzie, and M. Yung. Proactive RSA. In Advances in Cryptology—CRYPTO '97, volume 1294 of Lecture Notes in Computer Science, pages 440–454. Springer-Verlag, 17–21 Aug. 1997. 181, 182, 192
- 17. Y. Frankel, P. Gemmell, and M. Yung. Witness-based cryptographic program checking and robust function sharing. In STOC'96 [33], pages 499–508. 182
- Y. Frankel, P. D. MacKenzie, and M. Yung. Manuscript of current paper with complete proof. 190
- Y. Frankel, P. D. MacKenzie, and M. Yung. Adaptively-secure distributed publickey systems. In European Symposium on Algorithms—ESA '99, volume 1643 of Lecture Notes in Computer Science, pages 4–27. Springer-Verlag, 16–18 July 1998.
 182
- Y. Frankel, P. D. MacKenzie, and M. Yung. Robust efficient distributed rsa-key generation. In STOC'98 [34], pages 663–672. 185, 186, 193
- Y. Frankel and M. Yung. Distributed public-key cryptosystems. In H. Imai and Y. Zheng, editors, Advances in Public Key Cryptography—PKC '98, volume 1431 of Lecture Notes in Computer Science, pages 1–13. Springer-Verlag, Feb. 1998. invited talk. 180
- R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust and efficient sharing of RSA functions. In Advances in Cryptology—CRYPTO '96, volume 1109 of Lecture Notes in Computer Science, pages 157–172. Springer-Verlag, 18–22 Aug. 1996. 182
- S. Goldwasser. Multi-party computations: Past and present. In Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, pages 1–6, 1997. invited talk. 180
- A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public-key and signature schemes. In *Proceedings of the Third Annual Conference* on Computer and Communications Security, pages 100–110, 1996.
- A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing, or: How to cope with perpetual leakage. In Advances in Cryptology—CRYPTO '95, volume 963 of Lecture Notes in Computer Science, pages 339–352. Springer-Verlag, 27–31 Aug. 1995. 183

- J. Kilian, E. Petrank, and C. Rackoff. Lower bounds for zero knowledge on the internet. In 39th Annual Symposium on Foundations of Computer Science, pages 484–492. IEEE, Nov. 1998. 193
- H. Krawczyk, editor. Advances in Cryptology—CRYPTO '98, volume 1462 of Lecture Notes in Computer Science. Springer-Verlag, 17–21 Aug. 1998. 191, 193
- 28. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proceedings* of the Tenth Annual ACM Symposium on Principles of Distributed Computing, pages 51–61, 1991. 180, 181, 183
- T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In CRYPTO'91 [7], pages 129–140. 182, 185
- T. Rabin. A simplified approach to threshold and proactive rsa. In Krawczyk [27], pages 89–104. 181, 182
- 31. C. P. Schnorr. Efficient identification and signatures for smart cards. In CRYPTO'89 [6], pages 239–252. 193
- 32. A. Shamir. How to share a secret. Commun. ACM, 22:612-613, 1979. 185
- 33. Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, 22–24 May 1996. 190, 192
- 34. Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, Dallas, Texas, 23–26 May 1998. 191, 192

A Proofs

We use efficient ZK proofs of knowledge (POKs) derived from [20] and [5]. These are composed of combinations of Σ -protocols [4] (i.e., Schnorr-type proofs [31]). For each ZK proof that we need, we will have a separate "proof" protocol, but there will be a single "setup" protocol used for all ZK proofs. Say A wishes to prove knowledge of "X" to B. Then the setup protocol will consist of B making a commitment and proving that he can open it in a witness indistinguishable way [13], and the proof protocol will consist of A proving to B either the knowledge of "X" or that A can open the commitment. (See [5] for details.) This construction allows the proof protocols to be run concurrently without any timing constraints, as long as they are run after all the setup protocols have completed. (For more on the problems encountered with concurrent ZK proofs see [26,11,12].)

The (RSA-based) ZK-proof-setup protocols are exactly the Σ -protocols for commitments over q-one-way-group-homomorphisms (q-OWGH), given in [5]. Recall the q-OWGH for an RSA system with parameters (N,e) is $f(x)=x^e$ mod N (with q=e in this case).

Let KE denote the "knowledge error" of a POK.

We define ZKSETUP-RSA_{A,B}(N, e, g) as a protocol in which A generates a commitment C and engages B in a WH POK $(KE = 1/e)^7$ of (σ, σ') (with $\sigma \in Z_e$, $\sigma' \in Z_N^*$) where $C \equiv g^{\sigma}(\sigma')^e \mod N$.

We define ZKPROOF-DL-REP_{A,B}(N, e, m, g, h, E, D) as a protocol in which A, who knows integers $d \in (-a, a]$ and $d' \in (-b, b]$ such that $E \equiv m^d \mod$

This implies e must be exponentially large in the security parameter k in order to obtain a sound proof. However, if e is small (say e=3) we can use different setup and proof protocols described in [5] to obtain provably secure and robust RSA-based protocols.

N and $D \equiv g^d h^{d'} \mod N$, engages B in a WH POK (KE = 1/e) of either $\Delta \in Z_e$, $\delta \in (-2ae(N+1), 2ae(N+1)]$, and $\delta' \in (-2be(N+1), 2be(N+1)]$ where $D^\Delta \equiv g^\delta h^{\delta'} \mod N$ and $E^\Delta \equiv g^\delta \mod N$, or (τ, τ') (with $\tau \in Z_e$, $\tau' \in Z_N^*$) where $C_{B,A} \equiv g^\tau(\tau')^e \mod N$ and $C_{B,A}$ is the commitment generated in ZKSETUP-RSA_{B,A}(N,e,g). This protocol is honest-verifier statistical zero-knowledge with a statistical difference between the distribution of views produced by the simulator and in the real protocol bounded by 2/N.

A.1 Proof of Representations

Here we give the main Σ -protocol used in ZKPROOF-DL-REP_{A,B}(N, e, m, g, h, E, D).

- 1. Initially, the parameters (N, e, m, g, h, E, D) are public, and A knows integers $d \in (-a, a]$ and $d' \in (-b, b]$ such that $E \equiv m^d \mod N$ and $D \equiv g^d h^{d'} \mod N$.
- 2. A generates $r \in_R (-aeN, aeN]$ and $r' \in (-beN, beN]$, computes $V = m^r \mod N$ and $W = g^r h^{r'} \mod N$, and sends V, W to B.
- 3. B generates $c \in_R Z_e$ and sends c to A.
- 4. A computes z = cd + r and z' = cd' + r', and sends z, z' to B.
- 5. B checks that $m^z \equiv E^c V \mod N$ and $g^z h^{z'} = D^c W \mod N$.

In all steps, A and B also check that the values received are in the appropriate ranges.

The above is a POK of $\Delta \in Z_e$, $\delta \in (-2ae(N+1), 2ae(N+1)]$, and $\delta' \in (-2be(N+1), 2be(N+1)]$ in which $m^{\delta} \equiv E^{\Delta} \mod N$ and $g^{\delta}h^{\delta'} = D^{\Delta} \mod N$. The knowledge error is 1/e, and the protocol is honest-verifier statistical zero-knowledge, with a statistical difference between views produced by the simulator and those in the real protocol bounded by 2/N.

A.2 Security Proof Summary

We reduce the security of RSA to the security of our Proactive RSA protocol. Say there exists an adversary, after watching polynomially many messages signed and polynomially many update protocols run, can sign a new challenge message with non-negligible probability ρ . Then we will give a polynomial-time algorithm to break RSA with probability close to ρ . We run the adversary against a simulation of the protocol, and then present m^* to be signed. We will show that the probability that an adversary can distinguish the simulation from the real protocol is negligible, and thus the probability that it signs m^* is negligibly less than ρ .

A major tool that enables us to claim simulatability of secret-key function applications is the notion of a "faking server." The simulator exploits the "actions" of this server to assure that the view is simulatable while not knowing the secret key. This server can be chosen at random and its public actions are

⁸ Recall that this main protocol is combined with a Σ -protocol proving knowledge of a commitment generated in a setup protocol, using an "OR" construction.

indistinguishable from an honest server to the adversary. We have to backtrack the simulation **only if** the adversary corrupts this special server (which is a polynomial probability implying expected poly-time simulation).

Factorization of RSA-140 Using the Number Field Sieve*

Stefania Cavallar³, Bruce Dodson⁵, Arjen Lenstra¹, Paul Leyland⁶, Walter Lioen³, Peter L. Montgomery⁷, Brian Murphy², Herman te Riele³, and Paul Zimmermann⁴

Citibank, Parsippany, NJ, USA arjen.lenstra@citicorp.com

² Computer Sciences Laboratory, The Australian National University Canberra ACT 0200, Australia

murphy@cslab.anu.edu.au

³ CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands {cavallar, walter, herman}@cwi.nl

⁴ Inria Lorraine and Loria, Nancy, France Paul.Zimmermann@loria.fr

⁵ Lehigh University, Bethlehem, PA, USA bad0@Lehigh.edu

Microsoft Research Ltd., Cambridge, UK pleyland@microsoft.com

⁷ 780 Las Colindas Road, San Rafael, CA 94903–2346 USA Microsoft Research and CWI pmontgom@cwi.nl

Abstract. On February 2, 1999, we completed the factorization of the 140–digit number RSA–140 with the help of the Number Field Sieve factoring method (NFS). This is a new general factoring record. The previous record was established on April 10, 1996 by the factorization of the 130–digit number RSA–130, also with the help of NFS. The amount of computing time spent on RSA–140 was roughly twice that needed for RSA–130, about half of what could be expected from a straightforward extrapolation of the computing time spent on factoring RSA–130. The speed-up can be attributed to a new polynomial selection method for NFS which will be sketched in this paper.

The implications of the new polynomial selection method for factoring a 512-bit RSA modulus are discussed and it is concluded that 512-bit (= 155-digit) RSA moduli are easily and realistically within reach of factoring efforts similar to the one presented here.

1 Introduction

Factoring large numbers is an old and fascinating métier in number theory which has become important for cryptographic applications after the birth, in 1977, of

^{*} Paper 030, accepted to appear in the Proceedings of Asiacrypt '99, Singapore, November 14-18, 1999. URL: http://www.comp.nus.edu.sg/~asia99.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 195–207, 1999. © Springer-Verlag Berlin Heidelberg 1999

the public-key cryptosystem RSA [22]. Since then, people have started to keep track of the largest (difficult) numbers factored so far, and reports of new records were invariably presented at cryptographic conferences. We mention Eurocrypt '89 (C100¹ [14]), Eurocrypt '90 (C107 and C116 [15]), Crypto '93 (C120, [8]), Asiacrypt '94 (C129, [1]) and Asiacrypt '96 (C130, [6]). The 130-digit number was factored with help of the *Number Field Sieve* method (NFS), the others were factored using the *Quadratic Sieve* method (QS).

For information about QS, see [21]. For information about NFS, see [13]. For additional information, implementations and previous large NFS factorizations, see [9,10,11,12].

In this paper, we report on the factoring of RSA–140 by NFS and the implications for RSA. The number RSA–140 was taken from the RSA Challenge list [23]. In Sect. 2 we estimate how far we are now from factoring a 512–bit RSA modulus. In Sect. 3, we sketch the new polynomial selection method for NFS and we give the details of our computations which resulted in the factorization of RSA–140.

2 How far are we from factoring a 512-bit RSA modulus?

RSA is widely used today. We quote from RSA Laboratories' Frequently Asked Questions about today's Cryptography 4.0 (http://www.rsa.com/rsalabs/fag/html/3-1-9.html):

Question 3.1.9.
Is RSA currently in use?

RSA is currently used in a wide variety of products, platforms, and industries around the world. It is found in many commercial software products and is planned to be in many more. RSA is built into current operating systems by Microsoft, Apple, Sun, and Novell. In hardware, RSA can be found in secure telephones, on Ethernet network cards, and on smart cards. In addition, RSA is incorporated into all of the major protocols for secure Internet communications, including S/MIME (see Question 5.1.1), SSL (see Question 5.1.2), and S/WAN (see Question 5.1.3). It is also used internally in many institutions, including branches of the U.S. government, major corporations, national laboratories, and universities.

At the time of this publication, RSA technology is licensed by about 350 companies. The estimated installed base of RSA encryption engines is around 300 million, making it by far the most widely used public-key

¹ By "Cxxx" we denote a composite number having xxx decimal digits.

cryptosystem in the world. This figure is expected to grow rapidly as the Internet and the World Wide Web expand.

The best size for an RSA key depends on the security needs of the user and on how long the data needs to be protected. At present, information of very high value is protected by 512-bit RSA keys. For example, CREST [7] is a system developed by the Bank of England and used to register all the transfers of stocks and shares listed in the United Kingdom. The transactions are protected using 512-bit RSA keys. Allegedly, 512-bit RSA keys protect 95% of today's E-commerce on the Internet [24].

The amount of CPU time spent to factor RSA-140 is estimated to be only twice that used for the factorization of RSA-130, whereas on the basis of the heuristic complexity formula [3] for factoring large N by NFS:

$$\mathcal{O}\left(\exp\left((1.923 + o(1))(\log N)^{1/3}(\log\log N)^{2/3}\right)\right),$$

one would expect an increase in the computing time by a factor close to four. This has been made possible by algorithmic improvements (mainly in the polynomial generation step [18], and to a lesser extent in the sieving step and the filter step of NFS), and by the relative increase in memory speed of the workstations and PCs used in this project.

After the completion of RSA-140, we completely factored the 211-digit number $10^{211}-1$ with the Special Number Field Sieve (SNFS) at the expense of slightly more computational effort than we needed for RSA-140. We notice that the polynomial selection stage is easy for $10^{211}-1$. Calendar time was about two months. This result means a new factoring record for SNFS (see ftp://ftp.cwi.nl/pub/herman/NFSrecords/SNFS-211). The previous SNFS record was the 186-digit number $32633^{41}-1$ (see ftp://ftp.cwi.nl/pub/herman/NFSrecords/SNFS-186).

Experiments indicate that the approach used for the factorization of RSA–140 may be applied to RSA–155 as well. Estimates based on these experiments suggest that the total effort involved in a 512–bit factorization (RSA–155 is a 512–bit number) would require only a fraction of the computing time that has been estimated in the literature so far. Also, there is every reason to expect that the matrix size, until quite recently believed to be the main stumbling block for a 512–bit factorization using NFS, will turn out to be quite manageable. As a result 512–bit RSA moduli do, in our opinion, not offer more than marginal security, and should no longer be used in any serious application.

3 Factoring RSA-140

We assume that the reader is familiar with NFS [13], but for convenience we briefly describe the method here. Let N be the number we wish to factor, known to be composite. There are four main steps in NFS: polynomial selection, sieving, linear algebra, and square root.

In the polynomial selection step, two irreducible polynomials $f_1(x)$ and $f_2(x)$ with a common root $m \mod N$ are selected having as many as practically possible smooth values over a given factor base.

In the *sieving step* which is by far the most time-consuming step of NFS, pairs (a, b) are found with gcd(a, b) = 1 such that both

$$b^{\deg(f_1)}f_1(a/b)$$
 and $b^{\deg(f_2)}f_2(a/b)$

are smooth over given factor bases, i.e., factor completely over the factor bases. Such a pair (a, b) is called a *relation*. The purpose of this step is to collect so many relations that several subsets S of them can be found with the property that a product taken over S yields an expression of the form

$$X^2 \equiv Y^2 \pmod{N}. \tag{1}$$

For approximately half of these subsets, computing gcd(X - Y, N) yields a non-trivial factor of N (if N has exactly two distinct factors).

In the linear algebra step, the relations found are first filtered with the purpose of eliminating duplicate relations and relations in which a prime or prime ideal occurs which does not occur in any other relation. If a prime ideal occurs in exactly two or three relations, these relations are combined into one or two (respectively) so-called relation-sets. These relation-sets form the columns of a very large sparse matrix over \mathcal{F}_2 . With help of an iterative block Lanczos algorithm a few dependencies are found in this matrix. This is the main and most time- and space-consuming part of the linear algebra step.

In the square root step, the square root of an algebraic number of the form

$$\prod_{(a,b)\in S} (a-b\alpha)$$

is computed, where α is a root of one of the polynomials $f_1(x)$, $f_2(x)$, and where a, b and the cardinality of the set S are all a few million. The norms of all $(a - b\alpha)$'s are smooth. This leads to a congruence of the form (1).

In the next four subsections, we describe these four steps, as carried out for the factorization of RSA-140. We pay most attention to the polynomial selection step because, here, new ideas have been incorporated which led to a reduction of the expected – and actual – sieving time for RSA-140 (extrapolated from the RSA-130 sieving time) by a factor of 2.

3.1 Polynomial Selection

For number field sieve factorizations we use two polynomials $f_1, f_2 \in \mathbb{Z}[x]$ with, amongst other things, a common root $m \mod N$. For integers as large as RSA–140, a modified base-m method is the best method we know of choosing these polynomials. Montgomery's "two-quadratics" method [11] is the only known alternative, and it is unsuitable for numbers this large. With the base-m method,

we fix a degree d (here d=5) then seek $m \approx N^{1/(d+1)}$ and a polynomial f_1 of degree d for which

$$f_1(m) \equiv 0 \pmod{N}.$$
 (2)

The polynomial f_1 descends from the base-m representation of N. Indeed, we begin with $f_1(x) = \sum_{i=0}^d a_i x^i$ where the a_i are the coefficients of the base-m representation, adjusted so that $-m/2 \le a_i < m/2$.

Sieving occurs over the homogeneous polynomials $F_1(x,y) = y^d f_1(x/y)$ and $F_2(x,y) = x - my$. The aim for polynomial selection is to choose f_1 and m such that the values $F_1(a,b)$ and $F_2(a,b)$ are simultaneously smooth at many coprime integer pairs (a,b) in the sieving region.

We consider this problem in two stages; first we must decide what to look for, then we must decide how to look for it. The first stage requires some understanding of polynomial yield; the second requires techniques for generating polynomials with good yield. In this paper we seek only to outline our techniques. Full details will be published at a later date.

Polynomial Yield. The *yield* of a polynomial F(x, y) refers to the number of smooth (or almost smooth) values it produces in its sieve region. Ultimately of course we seek a *pair* of polynomials F_1 , F_2 with good yield. Since F_2 is linear, all primes are roots of F_2 , so the difficult polynomial is the non-linear F_1 . Hence, initially, we speak only of the yield of F_1 .

There are two factors which influence the yield of F_1 . These are discussed in a preliminary manner in [19]. We call the factors *size* and *root properties*. Choosing good F_1 requires choosing F_1 with a good *combination* of size and root properties.

By size we refer to the magnitude of the values taken by F_1 . It has always been well understood that size affects the yield of F_1 . Indeed previous approaches to polynomial selection have sought polynomials whose size is smallest (for example, [6]).

The influence of root properties however, has not previously been either well understood or adequately exploited. By root properties we refer to the extent to which the distribution of the roots of F_1 modulo small p^k , for p prime and $k \geq 1$, affects the likelihood of F_1 values being smooth. In short, if F_1 has many roots modulo small p^k , the values taken by F_1 "behave" as if they are much smaller than they actually are. That is, on average, the likelihood of F_1 -values being smooth is increased. We are able to exploit this property to the extent that F_1 values behave as if they are as little as 1/1000 their actual value. We estimate this property alone increases yield by a factor of four due (by comparison to sieving over random integers of the same size).

Generating Polynomials with good Yield. We consider this problem in two stages. In the first stage we generate a large sample of good polynomials. Although each polynomial generated has a good combination of size and root properties, there remains significant variation in the yield across the sample. Moreover, there are still far too many polynomials to conduct sieving experiments on each one. Thus in the second stage we identify *without* sieving, the best polynomials in the sample. The few polynomials surviving this process are then subjected to sieving experiments.

Consider the first stage. We concentrate on so-called *skewed* polynomials, that is, polynomials whose first few coefficients $(a_5, a_4 \text{ and } a_3)$ are small compared to m, and whose last few coefficients $(a_2, a_1 \text{ and } a_0)$ may be large compared to m. In fact usually $|a_5| < |a_4| < \ldots < |a_0|$. To compensate for the last few coefficients being large, we sieve over a region much longer in x than y. We take the region to be a rectangle whose length-to-height ratio is s.

Notice that any base-m polynomial may be re-written so that sieving occurs over a rectangle of skewness s. Let $m = O(N^{1/(d+1)})$ giving an unmodified base-m polynomial F_1 with coefficients also $O(N^{1/(d+1)})$. The expected sieve region for F_1 is a "square" given by $\{(x,y): -M \le x \le M \text{ and } 1 \le y \le M\}$ for some M. For some (possibly non-integer) $s \in \mathbb{R}$ let $x' = x/\sqrt{s}$, $y' = y\sqrt{s}$ and m' = ms. The polynomials $F_1(x',y')$ and $F_2(x',y')$ with common root m', considered over a rectangle of skewness s and area $2M^2$, have the same norms as F_1 and F_2 over the original square region. Such a skewing process can be worthwhile to increase the efficiency of sieving.

However, we have additional methods for constructing highly skewed polynomials with good yields. Hence, beyond simply skewing the region on unmodified base-m polynomials, we focus on polynomials which are themselves intrinsically skewed. The search begins by isolating skewed polynomials which are unusually small over a rectangle of some skewness s and which have better than average root properties. The first quality comes from a numerical optimization procedure which fits a sieve region to each polynomial. The second quality comes from choosing (small) leading coefficients divisible by many small p^k .

We then exploit the skewness to seek adjustments to f_1 which cause it to have exceptionally good root properties, without destroying the qualities mentioned above. We can make any adjustment to f_1 as long as we preserve (2). We make what we call a rotation by P for some polynomial P(x). That is, we let

$$f_{1,P}(x) = f_1(x) + P(x) \cdot (x - m)$$

where $P \in \mathbb{Z}[x]$ has degree small compared to d. Presently we use only linear $P(x) = j_1 x - j_0$ with j_1 and j_0 small compared to a_2 and a_1 respectively. We use a sieve-like procedure to identify pairs (j_1, j_0) which cause $f_{1,P}$ to have exceptionally good root properties mod small p^k . At the end of this procedure (with $p^k < 1000$ say) we have a large set of candidate polynomials.

Consider then the second stage of the process, where we isolate without sieving the polynomials with highest yield. Notice that as a result of looking at a large range of a_d the values of m may vary significantly across the sample. At this stage it is crucial then to consider both F_1 and F_2 in the rating procedure. Indeed, the values s vary across the sample too.

We use a quantitative estimate of the effect of the root properties of each polynomial. We factor this parameter into estimates of smoothness probabilities

for F_1 and F_2 across a region of skewness s. It is not necessary to estimate the yield across the region, simply to rank the polynomial pairs in the order in which we expect their yields to appear. Of course to avoid missing good polynomial pairs it is crucial that the metric so obtained be reliable.

At the conclusion of this procedure we perform short sieving experiments on the top-ranked candidates.

Results. Before discussing the RSA–140 polynomial selection results, we briefly consider the previous general factoring record, RSA–130 [6]. As a test, we repeated the search for RSA–130 polynomials and compared our findings to the polynomial used for the factorization. We searched for non-skewed polynomials only, since that is what was used for the RSA–130 factorization. Despite therefore finding fewer polynomials with exceptional root properties, we did, in a tiny fraction of the time spent on the RSA–130 polynomial search, find several small polynomials with good root properties. Our best RSA–130 polynomial has a yield approximately twice that of the polynomial used for the factorization. In essence, this demonstrates the benefit of knowing "what to look for".

The RSA–140 search however, further demonstrates the benefit of knowing "how to look for it". Here of course we exploit the skewness of the polynomials to obtain exceptional root properties.

Sieving experiments on the top RSA-140 candidates were conducted at CWI using line sieving. All pairs were sieved over regions of the same area, but skewed appropriately for each pair. Table 1 shows the relative yields of the top five candidate pairs, labeled A,..., E. These yields match closely the predictions of our pre-sieving yield estimate.

Poly.	Rel. Yield
Α	1.00
В	0.965
С	0.957
D	0.931
\mathbf{E}	0.930

Table 1. Relative Yields of the top RSA-140 polynomials

The chosen pair, pair A, is the following:

```
F_1(x,y) = \begin{array}{c} 43\,96820\,82840 & x^5 \\ +39031\,56785\,38960\,y\,\,x^4 \\ -7387\,32529\,38929\,94572\,y^2x^3 \\ -190\,27153\,24374\,29887\,14824\,y^3x^2 \\ -6\,34410\,25694\,46461\,79139\,30613\,y^4x \\ +31855\,39170\,71474\,35039\,22235\,07494\,y^5 \end{array}
```

and

$$F_2(x,y) = x - 34435657809242536951779007 y$$

with $s \approx 4000$.

Consider F_1 , F_2 with respect to size. We denote by a_{max} the largest $|a_i|$ for $i = 0, \ldots, d$. The un-skewed analogue, $F_1(63x, y/63)$, of $F_1(x, y)$ has

$$a_{max} \approx 5 \cdot 10^{20}$$
.

A typical unmodified base-m polynomial has

$$a_{max} \approx 1/2N^{1/6} \approx 8 \cdot 10^{22}$$
.

The un-skewed analogue, $F_2(63x, y/63)$, of $F_2(x, y)$ has

$$a_{max} \approx 3N^{1/6}$$
.

Hence, compared to the typical case F_1 values have shrunk by a factor about 160 whilst F_2 values have grown by a factor of 3. F_1 has real roots x/y near -4936, 2414, and 4633.

Now consider F_1 with respect to root properties. Notice that a_5 factors as $2^3 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 41 \cdot 29759$. Since also $4|a_4|$ and $2|a_3|$, $F_1(x,y)$ is divisible by 8 whenever y is even. $F_1(x,y)$ has at least three roots x/y modulo each prime from 3 to 17 (some of which are due to the factorization of the leading coefficient), and an additional 35 such roots modulo the 18 primes from 19 to 97.

We estimate that the yield of the pair F_1 , F_2 is approximately eight times that of a skewed pair of average yield. Approximately a factor of four in that eight is due to the root properties, the rest to its size. We estimate the effort spent on the polynomial selection to be equivalent to 0.23 CPU years (approximately 60 MIPS-years). Searching longer may well have produced better polynomials, but we truncated the search to make use of idle time on workstations over the Christmas period (for sieving). We leave as a subject of further study the trade-off between polynomial search time and the corresponding saving in sieving time.

3.2 Sieving

Partially for comparison, two sieving methods were used: lattice sieving and line sieving. The line siever fixes a value of y (from y = 1, 2, ... up to some bound) and finds values of x for which both $F_1(x, y)$ and $F_2(x, y)$ are smooth. The lattice siever fixes a prime q, called the special-q, which divides $F_1(x, y)$, and finds (x, y) pairs for which both $F_1(x, y)/q$ and $F_2(x, y)$ are smooth. This is carried out for many special-q's. Lattice sieving was introduced by Pollard [20] and the code we used is the implementation described in [12,6], with some additions to handle skew sieving regions efficiently.

For the lattice sieving, a rational factor base of 250 000 elements (the primes \leq 3 497 867) and an algebraic factor base of 800 000 elements (ideals of norm

 \leq 12 174 433) were chosen. For the line sieving, larger factor base bounds were chosen, namely: a rational factor base consisting of the primes < 8 000 000 and an algebraic factor base with the primes < 16 777 216 = 2^{24} . For both sieves the large prime bounds were 500 000 000 for the rational primes and 1 000 000 000 for the algebraic primes. The lattice siever allowed two large primes on each side, in addition to the special-q input. The line siever allowed three large primes on the algebraic side (this was two for RSA–130) and two large primes on the rational side.

The special-q's in the lattice siever were taken from selected parts of the interval [12 175 000, 91 000 000] and a total of 2 361 390 special-q's were handled. Lattice sieving ranged over a rectangle of 8192 by 4000 points per special-q, i.e., a total of about $7.7 \cdot 10^{13}$ points. Averaged over all the workstations and PCs on which the lattice siever was run, about 52 seconds were needed to handle one special-q and about 16 relations were found per special-q. So on average the lattice siever needed 3.25 CPU seconds to generate one relation.

Line sieving ranged over most of $|x| < 9\ 000\ 000\ 000$ and $1 \le y \le 70\ 000$, about $1.2 \cdot 10^{15}$ points. It would have been better to reduce the bound on x and raise the bound on y, in accordance with skewness 4000, but we overestimated the amount of line sieving needed. 30% of the relations found with the line-siever had three large primes. Averaged over all the workstations and PCs on which the line siever was run, it needed 5.1 CPU seconds to generate one relation.

A fair comparison of the performances of the lattice and the line siever is difficult for the following reasons: memory requirements of the two sievers are different; the efficiency of both sievers decreases – but probably not with the same "speed" – as the sieving time increases; the codes which we used for lattice and line sieving were optimized by different persons (Arjen Lenstra, resp. Peter Montgomery).

A total of 68 500 867 relations were generated, 56% of them with lattice sieving (indicated below by "LA"), 44% with line sieving (indicated by "LI").

Sieving was done at five different locations with the following contributions:

- 36.8 % Peter L. Montgomery, Stefania Cavallar, Herman J.J. te Riele, Walter M. Lioen (LI, LA at CWI, Amsterdam, The Netherlands)
- 28.8 % Paul C. Leyland (LA at Microsoft Research Ltd, Cambridge, UK)
- 26.6~% Bruce Dodson (LI, LA at Lehigh University, Bethlehem, PA, USA)
- 5.4 % Paul Zimmermann (LA at Médicis Center, Palaiseau, France)
- 2.5 % Arjen K. Lenstra (LA at Citibank, Parsippany, NJ, USA, and at the University of Sydney, Australia)

Sieving started the day before Christmas 1998 and was completed one month later. Sieving was done on about 125 SGI and Sun workstations running at 175 MHz on average, and on about 60 PCs running at 300 MHz on average. The total amount of CPU time spent on sieving was 8.9 CPU-years. We estimate this to be equivalent to 2000 MIPS years. For comparison, RSA–130 took about 1000 MIPS years. Practical experience we collected with factoring large RSA–numbers tells us that with a careful tuning of the parameters the sieving times

may be reduced now to 1000 resp. 500 MIPS years. The relations were collected at CWI and required 3.7 Gbytes of disk storage.

3.3 Filtering and Finding Dependencies

The filtering of the data and the building of the matrix were carried out at CWI and took one calendar week.

Filtering. Not all the sieved relations were used for filtering since we had to start the huge job for finding dependencies at a convenient moment. We actually used 65.7M of the 68.5M relations as filter input.

First, the "raw" data from the different contributing sites were searched through for duplicates. This single-contributor cleaning removed 1.4M duplicates. Next, we collected all the relations and eliminated duplicates again. This time, 9.2M duplicates were found. The $1.4\,+\,9.2\mathrm{M}$ duplicates came from machine and human error (e.g., the resumption of early aborted jobs resp. duplicate jobs), from the simultaneous use of the lattice and the line siever, and from the line siever and the lattice siever themselves.

In the filter steps which we describe next, we only considered prime ideals with norm larger than 10 million; in the sequel, we shall refer to these ideals as the large prime ideals. In the remaining 55.1M relations we counted 54.1M large prime ideals. We added 0.1M free relations (cf. [11, Sect. 4, pp. 234–235]). Taking into account another 1.3M prime ideals with norm below 10 million, it seemed that we did not have enough relations at this point. However, after we removed 28.5M so-called singletons (i.e., relations which contain a large prime ideal that does not appear in any other relation) we were left with 26.7M relations having 21.5M large prime ideals. So now we had more than enough relations compared with the total number of prime ideals. We deleted another 17.6M relations which were heuristically judged the least useful², or which became singletons after we had removed some other relations. We were finally left with 9.2M relations containing 7.8M large prime ideals. After this, relations with large prime ideals occurring twice were merged (6.0M relations left) and, finally, those occurring three times were merged (4.7M relations left).

Finding Dependencies. The resulting matrix had 4 671 181 rows and 4 704 451 columns, and weight 151 141 999 (32.36 nonzeros per row). With the help of Peter Montgomery's Cray implementation of the block Lanczos algorithm (cf. [17]) it took almost 100 CPU-hours and 810 Mbytes of central memory on the Cray C916 at the SARA Amsterdam Academic Computer Center to find 64 dependencies among the rows of this matrix. Calendar time for this job was five days.

 $[\]overline{}^{2}$ The criterion used for this filter step will be described in a forthcoming report [4].

3.4 The Square Root Step

During February 1–2, 1999, four square root (cf. [16]) jobs were started in parallel on four different 250 MHz processors of CWI's SGI Origin 2000, each handling one dependency. Each had about 5 million (not necessarily distinct) $a-b\alpha$ terms in the product. After 14.2 CPU-hours, one of the four jobs stopped, giving the two prime factors of RSA–140. Two others also expired with the two prime factors after 19 CPU-hours (due to different input parameter choices). One of the four jobs expired with the trivial factors.

We found that the 140-digit number

RSA-140 =

 $2129024631825875754749788201627151749780670396327721627823338321538194 \\ 9984056495911366573853021918316783107387995317230889569230873441936471$

can be written as the product of the two 70-digit primes:

p = 3398717423028438554530123627613875835633986495969597423490929302771479 and

q = 6264200187401285096151654948264442219302037178623509019111660653946049.

Primality of the factors was proved with the help of two different primality proving codes [2,5]. The factorizations of $p \pm 1$ and $q \pm 1$ are given by

 $2^3 3^2 5 13 \cdot 8429851 \cdot 33996935324034876299 \cdot 2534017077123864320746970114544624627539$

 $\begin{array}{l} q-1=\\ 2^661\cdot 135613\cdot 3159671789\cdot 3744661133861411144034292857028083085348933344798791\\ q+1= \end{array}$

 $2 \cdot 3 \cdot 5^2 389 \cdot 6781 \cdot 982954918150967 \cdot 16106360796654291745007358391328807590779968869$

Acknowledgements. Acknowledgements are due to the Dutch National Computing Facilities Foundation (NCF) for the use of the Cray C916 supercomputer at SARA, and to (in alphabetical order) CWI, Lehigh University, the Magma Group of John Cannon at the University of Sydney, the Médicis Center at École Polytechnique (Palaiseau, France), and Microsoft Research Ltd (Cambridge, UK), for the use of their computing resources.

References

 D. Atkins, M. Graff, A.K. Lenstra, and P.C. Leyland. THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE. In J. Pieprzyk and R. Safavi-Naini, editors, Advances in Cryptology – Asiacrypt '94, volume 917 of Lecture Notes in Computer Science, pages 265–277, Springer-Verlag, Berlin, 1995. 196

- Wieb Bosma and Marc-Paul van der Hulst. Primality proving with cyclotomy. PhD thesis, University of Amsterdam, December 1990. 205
- 3. J.P. Buhler, H.W. Lenstra, Jr., and Carl Pomerance. Factoring integers with the number field sieve. Pages 50–94 in [13]. 197
- 4. S. Cavallar. Strategies for filtering in the Number Field Sieve. In preparation. 204
- H. Cohen and A.K. Lenstra. Implementation of a new primality test. Mathematics of Computation, 48:103–121, 1987. 205
- James Cowie, Bruce Dodson, R.-Marije Elkenbracht-Huizing, Arjen K. Lenstra, Peter L. Montgomery, and Jörg Zayer. A world wide number field sieve factoring record: on to 512 bits. In Kwangjo Kim and Tsutomu Matsumoto, editors, Advances in Cryptology – Asiacrypt '96, volume 1163 of Lecture Notes in Computer Science, pages 382–394, Springer-Verlag, Berlin, 1996. 196, 199, 201, 202
- 7. CREST. Visit http://www.crestco.co.uk/. 197
- T. Denny, B. Dodson, A.K. Lenstra, and M.S. Manasse, On the factorization of RSA-120. In D.R. Stinson, editor, Advances in Cryptology - Crypto '93, volume 773 of Lecture Notes in Computer Science, pages 166-174, Springer-Verlag, Berlin, 1994. 196
- B. Dodson and A. K. Lenstra. NFS with four large primes: an explosive experiment. In D. Coppersmith, editor, Advances in Cryptology - Crypto '95, volume 963 of Lecture Notes in Computer Science, pages 372–385, Springer-Verlag, Berlin, 1995.
 196
- Marije Elkenbracht-Huizing. Factoring integers with the number field sieve. PhD thesis, Leiden University, May 1997. 196
- R.-M. Elkenbracht-Huizing. An implementation of the number field sieve. Experimental Mathematics, 5:231–253, 1996. 196, 198, 204
- R. Golliver, A.K. Lenstra, and K.S. McCurley. Lattice sieving and trial division. In Leonard M. Adleman and Ming-Deh Huang, editors, Algorithmic Number Theory, (ANTS-I, Ithaca, NY, USA, May 1994), volume 877 of Lecture Notes in Computer Science, pages 18–27, Springer-Verlag, Berlin, 1994. 196, 202
- A.K. Lenstra and H.W. Lenstra, Jr., editors. The Development of the Number Field Sieve, volume 1554 of Lecture Notes in Mathematics. Springer-Verlag, Berlin, 1993. 196, 197, 206, 207
- A.K. Lenstra and M.S. Manasse. Factoring by Electronic Mail. In J.-J. Quisquater and J. Vandewalle, editors, Advances in Cryptology – Eurocrypt '89, volume 434 of Lecture Notes in Computer Science, pages 355–371. Springer-Verlag, Berlin, 1990. 196
- 15. A.K. Lenstra and M.S. Manasse. Factoring with two large primes. In I.B. Dåmgard, editor, *Advances in Cryptology Eurocrypt '90*, volume 473 of *Lecture Notes in Computer Science*, pages 72–82. Springer-Verlag, Berlin, 1991. 196
- Peter L. Montgomery. Square roots of products of algebraic numbers. In Walter Gautschi, editor, Mathematics of Computation 1943–1993: a Half-Century of Computational Mathematics, pages 567–571. Proceedings of Symposia in Applied Mathematics, American Mathematical Society, 1994. 205
- Peter L. Montgomery. A block Lanczos algorithm for finding dependencies over GF(2). In Louis C. Guillou and Jean-Jacques Quisquater, editors, Advances in Cryptology Eurocrypt '95, volume 921 of Lecture Notes in Computer Science, pages 106–120, Springer-Verlag, Berlin, 1995. 204
- Peter L. Montgomery and Brian Murphy. Improved Polynomial Selection for the Number Field Sieve. Extended Abstract for the Conference on the Mathematics of Public-Key Cryptography, June 13–17, 1999, The Fields Institute, Toronto, Ontario, Canada. 197

- B. Murphy. Modelling the Yield of Number Field Sieve Polynomials. J. Buhler, editor, Algorithmic Number Theory, (Third International Symposium, ANTS-III, Portland, O, USA, June 1998), volume 1423 of Lecture Notes in Computer Science, pages 137–151, Springer-Verlag, Berlin, 1998. 199
- 20. J.M. Pollard. The lattice sieve. Pages 43–49 in [13]. 202
- Carl Pomerance. The Quadratic Sieve Factoring Algorithm. In T. Beth, N. Cot and I. Ingemarsson, editors, Advances in Cryptology Eurocrypt '84, volume 209 of Lecture Notes in Computer Science, pages 169–182, Springer-Verlag, New York, 1985. 196
- R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Comm. ACM, 21:120–126, 1978.
- 23. RSA Challenge Administrator. In order to obtain information about the RSA Factoring Challenge, send electronic mail to challenge-info@rsa.com and visit http://www.rsa.com/rsalabs/html/factoring.html. 196
- A. Shamir. Factoring Large Numbers with the TWINKLE device. Manuscript, April, 1999. 197

How to Prove That a Committed Number Is Prime

Tri Van Le¹, Khanh Quoc Nguyen², and Vijay Varadharajan²

¹ College of Engineering and Applied Science University of Wisconsin, Milwaukee EMS Building 3200 N Cramer Street, Milwaukee, WI 53201 lvtri@cs.uwm.edu
² School of Computing and Information Technology University of Western Sydney, Nepean P.O BOX 10 Kingswood, NSW 2747 Australia

{qnguyen, vijay}@cit.nepean.uws.edu.au

Abstract. The problem of proving a number is of a given arithmetic format with some prime elements, is raised in RSA undeniable signature, group signature and many other cryptographic protocols. So far, there have been several studies in literature on this topic. However, except the scheme of Camenisch and Michels, other works are only limited to some special forms of arithmetic format with prime elements. In Camenisch and Michels's scheme, the main building block is a protocol to prove a committed number to be prime based on algebraic primality testing algorithms. In this paper, we propose a new protocol to prove a committed number to be prime. Our protocol is O(t) times more efficient than Camenisch and Michels's protocol, where t is the security parameter. This results in O(t) time improvement for the overall scheme.

1 Introduction

In many applications, it is essential to prove that a number is of an arithmetic format of which some elements are prime. This problem is raised in many recently proposed cryptographic protocols [4,11,13,14]. The protocols proposed in [13,14] are sound only if there exists a proof that a given number n is a product of two safe prime numbers. In [11], the divisible electronic cash scheme requires a zero-knowledge proof that a committed number is a product of two primes. Furthermore, though not necessary, it is recommended in [15] to show that a number n is a product of two prime numbers p,q such that (p+1)/2 and (q+1)/2 are also primes.

Previously, there have been several studies in the literature related to this subject. de Graaf and Peralta [12] provided an efficient proof that a given number n is of the form $n = p^r q^s$, where r and s are odd, p and q are primes. Another protocol is that of Boyar et al.[1] which proves a given number n is square-free, i.e., all the factors of n are singular. Gennaro at el.[16] extended these two results to show that a number n is a product of quasi-safe primes p, q, i.e., each of (p-1)/2 and (q-1)/2 has only one prime factor.

More recently, Camenisch and Michels [3] proposed a general solution for this problem. They used the general paradigm of proving that a number is of a specific arithmetic format [6,8,10,17]. In this paradigm, the prover builds an arithmetic circuit corresponding to the arithmetic relation. She then commits all inputs of the circuit in some commitments. The proof is then a set of protocols showing that the prover knows the secret elements concealed in the commitments and the final output of the circuit is the desired number and the relations between committed elements correspond to the arithmetic circuit. As all elements are concealed in the commitments, in order to demonstrate that some elements are prime, the prover must be able to show that committed numbers are prime. In [3], a proof that a committed number is prime is at least $O(t^2)$ fold more expensive than a proof of an arithmetic relation.

Our main contribution of this paper is an efficient protocol to prove in (statistical) zero-knowledge that a committed number is prime. Our technique results in an efficient proof that a number is is of an arithmetic format where some involved elements are prime. The protocol is O(t) times more efficient than the protocol in [3], where $1/2^t$ is the error probability of the proof. This consequently leads to O(t) fold improvement of the general protocol.

2 Preliminary

In this section, we review a commitment scheme and statistical zero-knowledge proofs that demonstrate basic arithmetic relations amongst some commitments. The commitment scheme is unconditional hiding and conditionally binding and other protocols are statistical zero-knowledge. They are all well-known in literature. The reader is referred to [2,3,6,10,11] for detailed discussions of these protocols and other variations.

In the following, we assume that $G = \langle g \rangle$ is a group of large known order Q over the finite field \mathcal{Z}_P for some known prime P and h is a second generator of the group such that $\log_g h$ is not known to the prover.

A commitment scheme: To commit an element x, the prover chooses $r \in_R \mathcal{Z}_Q$ and sends $y = g^x h^r$ to the verifier. Given y, it is infeasible for the verifier to obtain any information about x and it is infeasible for the prover to find two different pairs (x,r) and (x',r') such that $y = g^x h^r = g^{x'} h^{r'}$ unless she can compute $\log_g(h)$.

Proving the knowledge of a representation: of the element y to the bases $g_1, ..., g_k$, involves proving the knowledge of $x_1, ..., x_k$ such that $y = \prod_{i=1}^k g_i^{x_i}$. The protocol works as follows. The prover chooses $r_1, ..., r_k \in_R \mathcal{Z}_Q$, computes $w := \prod_{i=1}^k g_i^{r_i}$, and sends w to the verifier. The verifier picks a random challenge $c \in_R \{0,1\}^t$ and sends it to the prover. The prover computes $s_i := r_i - cx_i \mod Q$ for i=1,...,t. The verifier accepts, iff $w = y^c \prod_{i=1}^k g_i^{s_i}$. Following the notations of [3,4], we denote this protocol as $PK\{(\alpha_1,...,\alpha_k): y = \prod_{i=1}^k g_i^{\alpha_i}\}$.

Proving the equality of discrete logarithm: to the bases g_1 and h_1 in the representation of elements y_1, y_2 to the bases (g_1, \ldots, g_k) and (h_1, \ldots, h_k) respectively, involves proving the knowledge of $x_1, \ldots, x_k, z_1, z_2, \ldots, z_k$ such that $x_1 = z_1, y_1 = \prod_{i=1}^k g_i^{x_i}$ and $y_2 = \prod_{i=1}^k h_i^{z_i}$. The protocol works as follows. The prover chooses $r_1, \ldots, r_k \in_R \mathcal{Z}_Q$ and $u_2, \ldots, u_k \in_R \mathcal{Z}_Q$, computes $w_1 := \prod_{i=1}^k g_i^{r_i}$ and $w_2 := h_1^{r_1} \prod_{i=2}^k h_i^{u_i}$, and sends w_1, w_2 to the verifier. The verifier picks a random challenge $c \in_R \{0,1\}^t$ and sends it to the prover. The prover computes $s_i := r_i - cx_i \mod Q$ for $i=1,\ldots,k$ and $v_i := u_i - cz_i \mod Q$ for $i=2,\ldots,k$. The verifier accepts iff $w_1 = y_1^c g_1^{s_1} \prod_{i=2}^k g_i^{s_i}$ and $w_2 = y_2^c h_1^{s_1} \prod_{i=2}^k h_i^{u_i}$. We denote this protocol as $\mathrm{PK}\{(\alpha_1,\ldots,\alpha_k,\beta_1,\ldots,\beta_k): \alpha_1 = \beta_1 \wedge y_1 = \prod_{i=1}^k g_i^{\alpha_i} \wedge y_2 = \prod_{i=1}^k h_i^{\beta_i}\}$.

Proving that a discrete logarithm is in a given range: This protocol proves that the discrete logarithm x of $y = g^x h^r$ satisfies $x \in [a, b]$ for given parameters a, b < Q/2. Several such schemes exist in literature. We review the scheme of [11] here. The protocol works as follows (e = |(b - a)/3| - 1):

- The prover chooses $x_1, r_1, r_2 \in_R [0, e]$, sets $x_2 = x_1 e$ and $w_1 := g^{x_1} h^{r_1}$ and $w_2 := g^{x_2} h^{r_2}$. She then sends the un-order pair (w_1, w_2) to the verifier.
- The verifier chooses $c \in_R [0,1]$ and sends c to the prover.
- If c = 0, the prover sends x_1, x_2, r_1, r_2 to the verifier. Otherwise, the prover sends $(x + x_i, r + r_i)$ (j = 1 or 2) such that $x + x_i \in [a + e, b e]$.
- The prover accepts iff $w_1 = g^{x_1}h^{r_1}$, $w_2 = g^{x_2}h^{r_2}$ when c = 0 and $yw_j = g^{x+x_j}h^{r+r_j}$ when c = 1.

This is repeated t times to achieve the error probability of $1/2^t$. We denote this protocol as $PK\{(\alpha_1, \ldots, \alpha_k) : y = \prod_{i=1}^k g_i^{\alpha_i} \wedge \alpha_1 \in [a, b]\}$. The scheme is not very efficient. Constructions of [5,10] are much more efficient but use a composite modulo m and require a proof that m is a product of two primes.

Building on these protocols, we next present zero-knowledge protocols to prove secret modular quadratic residue and secret modular exponentiation. Both protocols use a protocol that demonstrates secret modular multiplicative relation. The protocols to prove secret modular multiplicative and exponentiation relations, were introduced in [3]. We present them here for the sake of completeness.

Secret modular multiplicative relation: Assume that a prover has committed to x, y, z, n in the commitments c_x, c_y, c_z and c_n such that $0 < x, y, z, n < 2^l$ where l = |Q|/2 - 1. The prover can convince the verifier that $xy \equiv z \mod n$ using the following proofs:

- (1) $PK\{(x, r_x) : c_x = g^x h^{r_x} \land x \in [1, 2^l]\}.$
- (2) $PK\{(y, r_y) : c_y = g^y h^{r_y} \land y \in [1, 2]\}.$
- (3) $PK\{(z, r_z) : c_z = g^z h^{r_z} \land z \in [1, 2^l]\}.$
- (4) $PK\{(n, r_n) : c_n = g^n h^{r_n} \land n \in [1, 2^l]\}.$
- (5) $PK\{(u, r_u) : c_u = g^u h^{r_u} \land u \in [1, 2^l]\}.$

(6)
$$PK\{(y, u, r_y, r_u, \rho) : c_z = c_x^y c_n^u h^\rho \wedge c_y = g^y h^{r_y} \wedge c_u = g^u h^{r_u} \}$$

Here clause (6) is a combination of two proofs of equality of discrete logarithms [4]. It is straightforward to prove that clause (6) is a zero-knowledge proof of knowledge. We denote this protocol as $PK\{(x, y, z, n) : xy \equiv z \mod n\}$.

Lemma 1. Let x, y, z, n be the values committed in c_x, c_y, c_z and c_n respectively. $PK\{(x, y, z, n) : xy \equiv z \mod n\}$ is a statistical zero-knowledge proof of $xy \equiv z \mod n$.

Proof. The statistical zero-knowledge claim follows from the statistical zero-knowledge property of the protocol components. We now show why the multiplicative relation holds.

We let the knowledge extractor to run the protocol with the prover. From (1)(2)(3) (4) and (5), the knowledge extractor can obtain $\tilde{x}, \tilde{y}, \tilde{z}, \tilde{n}, \tilde{u}\tilde{r}_x, \tilde{r}_y, \tilde{r}_z, \tilde{r}_n$ and \tilde{r}_u such that $c_x = g^{\tilde{x}}h^{\tilde{r}_x}$, $c_y = g^{\tilde{y}}h^{\tilde{r}_y}$, $c_z = g^{\tilde{z}}h^{\tilde{r}_z}$, $c_n = g^{\tilde{n}}h^{\tilde{r}_n}$ and $c_u = g^{\tilde{u}}h^{\tilde{r}_u}$. Moreover $0 < \tilde{x}, \tilde{y}, \tilde{z}, \tilde{n}, \tilde{u} < 2^l$.

Furthermore from (6), the extractor can extract $\tilde{\rho}, \tilde{y}, \tilde{u}, r_{\tilde{y}}$ and $\tilde{r_u}$ such that $c_z = c_x^{\tilde{y}} c_n^{\tilde{u}} h^{\tilde{\rho}}$, $c_y = g^{\tilde{y}} h^{\tilde{r_y}}$ and $c_u = g^{\tilde{u}} h^{\tilde{r_u}}$. Assuming that $\log_g(h)$ is not known, this shows $\tilde{z} = \tilde{x}\tilde{y} + \tilde{u}\tilde{n} \mod Q$. But $0 < \tilde{x}, \tilde{y}, \tilde{z}, \tilde{n}, \tilde{u} < 2^l$ and l < |Q|/2. Hence $\tilde{z} = \tilde{x}\tilde{y} + \tilde{u}\tilde{n} \mod Q$ holds only if $\tilde{z} = \tilde{x}\tilde{y} + \tilde{u}\tilde{n}$ holds, i.e., $\tilde{z} = \tilde{x}\tilde{y} \mod \tilde{n}$ holds for the committed values \tilde{x}, \tilde{y} and \tilde{n} .

Secret modular quadratic residue: Using the proof of secret modular multiplicative relation, the prover can prove that x is a quadratic modulo n for x and n committed in c_x and c_n respectively using $PK\{(y, y, x, n) : y^2 \equiv x \mod n\}$. This is because if there exists y such that $y^2 \equiv x \mod n$, then x is a quadratic residue modulo n. Let us denote this protocol as $PK\{(x, n) : x \in QR_n\}$.

Secret modular exponentiation relation: Given the commitments c_x, c_y, c_z and c_n , to prove that $x^y \equiv z \mod n$, the prover proceeds as follows:

- Let $y = \sum_{i=0}^{l-1} y_i 2^i$, $(y_i \in [0,1])$ and $x_0 = x$, $x_i = x_{i-1}^2 \mod n$ $(i = 1, \dots, l-1)$. Also let $u_i = x_i^{y_i}$ and $w_i = w_{i-1}u_i \mod n$ $(i = 0, \dots, l-1)$ and $w_0 = 1$.
- The prover commits to all x_i, y_i, u_i, w_i (i = 1, ..., l-1) in the commitments:

$$c_{y_i} = g^{y_i} h^{\hat{r}_i}$$

$$c_{x_i} = g^{x_i} h^{\hat{r}_i} \qquad (c_{x_0} = c_x)$$

$$c_{u_i} = g^{u_i} h^{\hat{r}_i}$$

$$c_{w_i} = g^{w_i} h^{\bar{r}_i} \qquad (c_{w_{l-1}} = c_z).$$

She then sends all her commitments to the verifier.

- The prover and the verifier now engage in the following protocols $(i = 0, \ldots, l-2)$.
 - (1) $PK\{(x_i, x_i, x_{i+1}, n) : x_i^2 \equiv x_{i+1} \mod n\}$
 - (2) $PK\{(w_i, u_{i+1}, w_{i+1}, n) : w_i u_{i+1} \equiv w_{i+1} \mod n\}$

- (3) $PK\{(\omega): (\prod_{i=0}^{l-1} c_{y_i}^{2^i})/c_y \equiv h^{\omega} \mod Q\}$ (4) $PK\{(x_i, y_i, u_i): y_1 \in [0, 1] \land u_i \equiv x_i^{y_i}\}$ using the sub-protocol described below.
- (5) $PK\{(w_0): w_0 \equiv 1\}$

Let us denote this protocol as $PK\{(x, y, z, n) : x^y \equiv z \mod n\}$. The intuition is that clause (1) shows that $x_{i+1} = x_i^2 \mod n$, $\forall i = 0, \ldots, l-2$. Because of $c_{x_0} =$ c_x , we have $x_0 = x$ and thus $x_i = x^{2^i}$. Next clause (4) shows $u_i \equiv x_i^{y_i}$. Hence clauses (1) and (4) show $u_i = x^{y_i 2^i} \mod n$. Furthermore, clauses (2) and (5) show that $w_{i+1} = w_i u_{i+1} \mod n$ $(i = 0, \dots, l-2)$ and $w_0 = 1$, this implies that $w_i = 0$ $\prod_{j=0}^{i} u_i = \prod_{j=0}^{i} (x^{y_j 2^j}) \mod n$. This further implies that $w_{l-1} = \prod_{j=0}^{l-1} x^{y_j 2^j} = \prod_{j=0}^{l-1} x^{y_j 2^j}$ $x^{\sum_{j=0}^{l-1} y_j 2^j} \mod n$. However clause (3) shows that the discrete logarithms of c_y and $(\prod_{i=0}^{l-1} c_{y_i}^{2^i})$ to base g are equivalent, i.e., $y = \sum_{j=0}^{l-1} y_j 2^j$. Thus it is clear that $w_{l-1} = x^{\sum_{j=0}^{l-1} y_j 2^j} \mod n = x^y = \mod n$. Finally, as $c_{w_{l-1}} = c_z$ and commitments are conditionally binding, we have $z = w_{l-1} = x^y \mod n$.

Now it remains to show the existence of the sub-protocol.

Sub-protocol Given three commitments c_{x_i}, c_{y_i} and c_{u_i} , the sub-protocol proves that $y_i \in [0,1], u_i = x_i^{y_i}$. Because $y_i = 0$ or 1, we only have to consider two

- 1. Case 1: $y_i = 0$. We have $c_{y_i} = h^{\hat{r}_i}$ for some \hat{r}_i . Also $u_i = x_i^{y_i}$ iff $c_{u_i} = gh^{\check{r}_i}$. This is equivalent to showing that $c_{u_i}/g = h^{r_i}$ for some r_i .
- 2. Case 2: $y_i = 1$. This means $c_{y_i} = gh^{\hat{r}_i}$ or $c_{y_i}/g = h^{\hat{r}_i}$. Also now we have $c_{u_i} = g^{x_i} h^{\check{r}_i}$ or $c_{u_i}/c_{x_i} = h^{r_i}$ for some r_i .

Thus to show that $u_i = x_i^{y_i}$, one has to show the knowledge of:

$$(c_{y_i} = h^{\hat{r}_i} \wedge c_{u_i}/g = h^{r_i}) \vee (c_{y_i}/g = h^{\hat{r}_i} \wedge c_{u_i}/c_{x_i} = h^{r_i}).$$

For clarity, we present the proof for:

$$(\alpha = h^{r_{\alpha}} \wedge \beta = h^{r_{\beta}}) \vee (\eta = h_{\eta}^{r} \wedge \kappa = h^{r_{\kappa}}).$$

Without loss of generality, we assume that the prover knows the $\log_h \alpha$ and $\log_b \beta$. The protocol works as follows:

- The prover chooses $\rho_1, \sigma_2, \mu_2, \lambda_2$, computes $\psi_1 := h^{\rho_1}, \psi_2 := h^{\sigma_2} \eta^{\mu_2} \kappa^{\lambda_2}$ and sends ψ_1, ψ_2 to the verifier.
- The verifier chooses random $\lambda, \mu \in \mathcal{Z}_Q$. He sends (λ, μ) to the prover.
- The prover computes $\mu_1 := \mu \oplus \mu_2$, $\lambda_1 := \lambda \oplus \lambda_2$ and $\sigma_1 := \rho_1 \mu_1 r_\alpha \lambda_1 r_\beta$. She then sends $(\sigma_1, \mu_1, \lambda_1, \sigma_2, \mu_2, \lambda_2)$ to the verifier.
- The verifier accepts iff $\mu = \mu_1 \oplus \mu_2$, $\lambda = \lambda_1 \oplus \lambda_2$, $\psi_1 = h^{\sigma_1} \alpha^{\mu_1} \beta^{\lambda_1}$ and $\psi_2 = h^{\sigma_2} \eta^{\mu_2} \kappa^{\lambda_2}$.

This is an example of zero-knowledge proof of arbitrary monotonic statements built with \wedge 's and \vee 's. In this protocol, \oplus denotes the XOR operation. Such proofs are discussed in [2,7]. For this reason, its security proof is omitted here. The reader is referred to [2,7] for further discussions.

3 Main Result

Our main result is an efficient zero-knowledge proof of a committed number n to be prime. The proof consists of two steps. First we show that n has only one prime factor. Next we show that n is square free. Clearly then n must be prime. We assume that that $2\tau \geq |n| \geq 2t$ for some known τ and t, which is also the security parameter. This can be proven using the protocol that proves a discrete logarithm is in a given range described earlier.

3.1 Proving That n Has Only One Prime Factor

Given an odd prime number n committed in a commitment commit(n), this subsection presents a statistical zero-knowledge protocol which convinces the prover that n has only one prime factor. First we need to show that n is odd. This is done as follows:

- PK $\{k, 2, n-1, 2^{2\tau+1}: 2k \equiv n-1 \mod 2^{2\tau+1}\}$, where commit(n-1) is computed as commit(n)/g.
- $-2^{2t-1} \le k \le 2^{2\tau}$ with the proof of a discrete logarithm in a given range.

As $n-1 < 2^{2\tau+1}$ and $\mathsf{commit}(n-1)/\mathsf{commit}(n) = g$, the proof demonstrates that $2k \equiv n-1$. Next we show that n has only one prime factor. There are two different methods of proving that. One works for the case $n \equiv 3 \mod 4$. The other works for any odd n. The former is more efficient. So far $n \equiv 3 \mod 4$ shows no apparent security weakness. In fact it is recommended in many applications(e.g. in Blum numbers) to choose prime numbers of this form. We present both methods here.

Specific case $n \equiv 3 \mod 4$. The proof that n has only a prime factor, is the following protocol:

- COMMON INPUT: a commitment commit(n) of a prime number n satisfying $n \equiv 3 \mod 4$.
- Repeat t times:
 - RANDOM INPUT: $0 < x < 2^{\tau}$
 - Prover: outputs a quadratic residue z modulo n out of $\pm x$, i.e, z = x or -x, a commitment commit(y) of the square root $y = z^{1/2} \mod n$) and proves that z is quadratic residue modulo n using $PK\{(z, n) : z \in QR_n\}$.
- Verifier: accepts iff he accepts all t proofs.

The zero-knowledge property of the protocol comes from the statistical zero-knowledge of involved proofs. We do not further evaluate them. Here we prove the soundness and completeness of the protocol. Before progressing further, let us review some basic number theory facts [9]:

1. For any odd prime number n, (-1) is a quadratic non-residue modulo n if and only if $n \equiv 3 \mod 4$.

- 2. Let n be an odd prime number. For any values u and v, uv is a quadratic residue modulo n if and only if either both or none of u and v are quadratic residue.
- 3. From (2), we can derive that for an odd prime n and a quadratic non-residue u, only one of v or uv is quadratic residue modulo n for any given v.
- 4. If n has more than one prime factor, a random number x is quadratic residue with no better than 1/4 probability.

Next to the proof of completeness and soundness.

- Completeness: Because n is an odd prime and $n \equiv 3 \mod 4$, (-1) is a quadratic non-residue modulo n. This means that out of $\pm x$, there is one and only one quadratic residue modulo n. The protocol completeness follows.
- **Soundness:** Observe that if an odd number n has more than one prime factor, then for a random non-zero number, the probability that it is quadratic modulo n is at most 1/4. If (x) is a quadratic non-residue, from [16], we have that (-x) is a quadratic residue is with the probability of 1/2. Thus the error probability of each round is 1/2. After t rounds, the error probability is $1/2^t$.

General case. The proof that n has only one prime factor, is based on the following protocol:

- COMMON INPUT: n an odd prime.
- Repeat 24t times:
 - RANDOM INPUT: $0 < x < 2^{\tau}$
 - Prover: either says that x is quadratic non-residue or runs $PK\{(x, n) : x \in QR_n\}$ to prove that x is quadratic.
- Verifier: accepts iff he accepts at least 9t proofs.

The zero-knowledge property is straightforward. The completeness and soundness intuition is as follows. To convince the verifier that n is prime, clearly the prover must try to show as many random input x's as possible are in QR_n . Observe that the probability of a random $x \in QR_n$ is 1/2 if n is prime and at most 1/4 if n has 2 or more prime factors. Thus ideally, out of 24t random x's, there should be 12t quadratic residues if n is prime and at most 6t quadratic residues if n has more than one prime factor. Using elementary probability theory, we have the following lemmas (see appendix for the proof of the lemmas):

Lemma 2. For a number $t \ge 40$, the probability that there exists 9t quadratic residues modulo n out of 24t random numbers is at least $1 - 1/2^t$ if a random number is quadratic residue with the probability of 1/2.

Lemma 3. For a number $t \geq 40$, the probability that there exists 9t quadratic residues modulo n out of 24t random numbers is at most $1/2^t$ if a random number is quadratic residue with the probability of 1/4.

Subsequently, we choose the threshold 9t so that the error probability of the protocol (i.e. the probability of failure for the honest prover, and the probability of success for the dishonest prover) is at most $(1/2^t)$ where the value of t is assumed to be at least 40.

3.2 Proving That n Is Square-Free

In this step, we can safely assume that n has only an odd prime factor, i.e., $n = p^{\alpha}$ for a prime p and $\alpha \ge 1$. In order to prove that n is square-free, the prover and the verifier runs the following protocol:

- RANDOM INPUT: $0 < x < 2^{\tau}$
- Prover: runs the proof $PK\{(x, n, x, n) : x^n \equiv x \mod n\}$ to show $x^n \equiv x \mod n$.
- Verifier: accepts that n is square-free if he accepts the proof $PK\{(x, n, x, n) : x^n \equiv x \mod n\}$

Again, the zero-knowledge property of the protocol comes from the statistical zero-knowledge of the associated proofs. We do not evaluate them further. The completeness is straightforward. It remains to show the soundness of the protocol.

Theorem 4. Assume that n has only a prime factor, then the protocol proves that n is square-free and so prime, with overwhelming probability.

Proof. Let $n = p^{\alpha}$, where p is the prime factor of n. To prove the theorem, we show that if $\alpha > 1$, $x^n \equiv x \mod n$ happens with negligible probability for a randomly chosen x.

First, consider the case $gcd(x,n) \neq 1$. As $n = p^{\alpha}$, p divides gcd(x,n). This implies that p^{α} divides x^{α} . But since x^{α} divides x^{n} , we have n divides x^{n} . Thus $x^{n} \equiv x \mod n$ is equivalent to $x \equiv 0 \mod n$ which happens with negligible probability for $0 < x < 2^{\tau}$.

Now we can conclude that $x^n \equiv x \mod n$ occurs with non-negligible probability only if gcd(x,n)=1. This means that $x\in Z_n^*$, where Z_n^* denotes the set of all numbers in Z_n relatively prime to n. The order of Z_n^* is $\phi(n)=(p-1)p^{\alpha-1}$. So for $x\in Z_n^*$, $x^{n-1}\equiv 1 \mod n$ holds only if $g^{gcd(n-1,\phi(n))}=g^{p-1}=1$. As the order of Z_n^* is $(p-1)p^{\alpha-1}$, there are only (p-1) such x's in Z_n^* . Hence for a random $x\in Z_n^*$. the probability that $x^n\equiv x \mod n$ is $(p-1)/(p-1)p^{\alpha-1}=1/p^{\alpha-1}$ which is negligible if $\alpha>1$

3.3 Efficiency Comparison with Previous Works

We consider a basic proof of knowledge of secret modular multiplicative relation as the basic proof. Each secret modular exponentiation relation proof is estimated to cost about 3t basic proofs.

The only other general protocol is that of Camenisch and Michels [3] requires t modular exponentiation relation proof. This is equivalent to about $3t^2$ basic proofs.

In our more efficient but less general version, the first step which proves n to have one prime factor uses t secret modular quadratic residue proofs. The second step is in fact a proof of secret modular exponentiation relation, which uses 3t basic proofs. Thus our total computation and communication costs is 4t basic proofs. This means an improvement of the order of 0.75(t).

In the more general but less efficient version, the first step requires 24t independent procedures. In each procedure, the prover either says a random number is a quadratic non-residue or proves that the number is a quadratic residue. The cost of saying that a number is a quadratic non-residue, is negligible. The cost of proving a quadratic residue is equivalent to a basic proof. Of course the verifier can always accept the proof once 9t proofs of quadratic residuosity are achieved. Thus in practice, the first step costs 9t basic proofs. The second step which is the same for both of our protocols, requires 3t basic proofs. Hence on average, the protocol costs 12t proofs. This means that the gained efficiency over the protocol of [3] is of the order of 0.33(t).

In practice, if t = 40, our two protocols are about an order of 30 and 12 times respectively more efficient than the protocol of [3]. For the case t = 80, the figures are about 60 and 25 times, respectively.

3.4 Generating a Random Number x

In both steps, the protocol makes use of some random numbers x's. In case such random numbers do not exist, a random number x can be generated as follows:

- The prover chooses a random x_1 , commits it in the commitment c_{x_1} and sends it to the verifier.
- The verifier chooses a random x_2 , commits it in the commitment c_{x_2} and sends it to the prover.
- The prover opens the commitment c_{x_1} and sends x_1 to the verifier.
- The verifier opens the commitment c_{x_2} and sends x_2 to the prover.
- If x_1 is consistent with c_{x_1} and x_2 is consistent with c_{x_2} , then the random number x is computed as $x = x_1 + x_2 \mod 2^{|N|}$.

This technique is known to be secure. The reader is referred to [3,16] for further details.

References

- J. Boyar, K. Friedl and C. Lund, Practical zero-knowledge proofs: giving hints and using deficiencies, Journal of Cryptology, 4(3):185-206, 1991.
- S. Brands, "Rapid Demonstration of Linear Relations Connected by Boolean Operators", Proceedings of Eurocrypt'97, LNCS 1223, pp. 318-333. 209, 212, 212
- J. Camenisch and M. Michels, "Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes", Proceedings of Eurocrypt'99, LNCS 1592, pp. 106–121. Also appeared as BRICS Technical Report RS-98-29. 209, 209, 209, 209, 209, 210, 215, 216, 216, 216
- J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups", Proceedings of CRYPTO '97, LNCS 1294, pages 410-424.
 208, 209, 211
- A. Chan, Y. Frankel and T. Tsiounis, "Easy come-easy go divisible cash", Proceedings of Eurocrypt'98, LNCS, pp. 561-575.

- R.Cramer and I. Damgard, "Zero-knowledge for Finite Field Arithmetic or: Can Zero-knowledge be for Free?", In Proceedings of CRYPTO '98, LNCS 1462, pp. 424–441, 1998. 209, 209
- R.Cramer, I.Damgard and B.Schoenmakers, Crypto'94, "Proofs of partial knowledge and simplified design of witness hiding protocols", Proceedings of CRYPTO '94, LNCS 839, pp.174-187.
- A. De Santis, G. Di Crescenzo and G. Persiano, "Secret sharing and perfect zero-knowledge", Proceedings of CRYPTO' 93, LNCS 773, pp.73-84. 212, 212 209
- A. Menezes, P. van Oorschot and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.
- E. Fujisaki and T. Okamoto, "Statistical zero-knowledge protocols to prove modular polynomial relation", Proceedings of CRYPTO '97, LNCS 1294, pp. 16-30. 209, 209, 210
- T. Okamoto An efficient divisible electronic cash scheme, Proceedings of CRYPTO '95, LNCS, pp. 439-451. 208, 208, 209, 210
- J. van de Graaf and R. Peralta, "A simple and secure way to show the validity of your public key", Proceedings of CRYPTO '87, LNCS 293, pp. 128-134.
- R. Gennaro, H. Krawczyk and T. Rabin, "RSA-based undeniable signatures", Proceedings of CRYPTO '97, LNCS 1294, pp. 132-149. 208, 208
- R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, "Robust and efficient sharing of RSA functions", Proceedings of CRYPTO '96, LNCS 1109, pp. 157-172. 208, 208
- 15. K.Koyama, U. Maurer, T. Okamoto and S. Vanstone, "New public-key schemes based on elliptic curves over the ring \mathbb{Z}_n ", Proceedings of CRYPTO '91, pp.252-266 208
- R. Gennaro, D. Micciancio and T. Rabin, "An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products", in Proceedings of 5rd ACM conference on Computer and Communication Security, 1998. 208, 214, 216
- M. Stadler, "Cryptographic Protocols for Revocable Privacy", Ph.D Thesis, Swiss Federal Institute of Technology, Zurich 1996.

A Proof of Lemma 2

Given a random number that is quadratic residue with the probability of 1/2, the probability that there are exact i quadratic residues in 24t random numbers is $\frac{1}{2^{24t}}\binom{24t}{i}$. Thus the probability that there are at least 9t quadratic residues in 24t random numbers is $\mathcal{P} = \sum_{i=9t}^{24t} \frac{1}{2^{24t}}\binom{24t}{i}$. We then have $\mathcal{P} = \sum_{i=9t}^{24t} \frac{1}{2^{24t}}\binom{24t}{i} > \sum_{i=12t}^{21t} \frac{1}{2^{24t}}\binom{24t}{i} = \sum_{i=0}^{9t} \frac{1}{2^{24t}}\binom{24t}{i+3t}$.

Furthermore, since $1 = \sum_{i=0}^{24t} \frac{1}{2^{24t}} {24t \choose i}$, we have $1 - \mathcal{P} = \sum_{i=0}^{9t-1} \frac{1}{2^{24t}} {24t \choose i}$.

Furthermore for $0 \le i < 9t$, we have

$$\frac{\binom{24t}{i+3t}}{\binom{24t}{i}} = \frac{\frac{24t!}{(3t+i)!(24t-3t-t)!}}{\frac{24t!}{4!(24t-i)!}} = \frac{(24t-i)\dots(24t-3t-i)}{(3t+i)\dots i}.$$

As $0 \le i < 9t$,

$$\frac{\binom{24t}{i+3t}}{\binom{24t}{i}} \ge \frac{(24t-9t)\dots(24t-3t-9t)}{(3t+9t)\dots(9t)} = \frac{(15t)\dots(12t)}{(12t)\dots(9t)} > \left(\frac{15}{12}\frac{12}{9}\right)^{3t/2} > 2^t.$$

So we now have

$$\mathcal{P} > \sum_{i=0}^{9t} \frac{1}{2^{24t}} \binom{24t}{i+3t} > 2^t \sum_{i=0}^{9t} \frac{1}{2^{24t}} \binom{24t}{i} = 2^t (1-\mathcal{P})$$

This means $\mathcal{P} > 1 - 1/2^t$ which completes the proof of lemma 2.

B Proof of Lemma 3

Given a random number that is quadratic residue with the probability of 1/4, the probability that there are exact i quadratic residues in 24t random numbers is $\frac{3^{24t-i}}{4^{24t}}\binom{24t}{i}$. Thus the probability that there are at least 9t quadratic residues in 24t random numbers is $\mathcal{P} = \sum_{i=9t}^{24t} \frac{3^{24t-i}}{4^{24t}}\binom{24t}{i}$. We then have $\mathcal{P} = \sum_{i=9t}^{24t} \frac{3^{24t-i}}{4^{24t}}\binom{24t}{i} < 5\sum_{i=9t}^{12t} \frac{3^{24t-i}}{4^{24t}}\binom{24t}{i} = 5\sum_{i=6t}^{9t} \frac{3^{21t-i}}{4^{24t}}\binom{24t}{i+3t}$.

Further, since $1 = \sum_{i=0}^{24t} \frac{3^{24t-i}}{4^{24t}} {2tt \choose i}$, we have

$$1 - \mathcal{P} = \sum_{i=0}^{9t-1} \frac{3^{24t-i}}{4^{24t}} \binom{24t}{i} > \sum_{i=6t}^{9t} \frac{3^{24t-i}}{4^{24t}} \binom{24t}{i}.$$

As $6t \le i < 9t$ and $t \ge 40$,

$$\frac{\frac{3^{24t-i}}{4^{24t}}\binom{24t}{i}}{\frac{3^{21t-i}}{4^{24t}}\binom{24t}{i+3t}} = 3^{3t}\frac{\binom{24t}{i}}{\binom{24t}{i+3t}} = 3^{3t}\frac{(3t+i)\dots(i)}{(24t-i)\dots(21t-i)} \ge 3^{3t}\frac{(9t)\dots(6t)}{(18t)\dots(15t)} > 5(2^t).$$

So we now have

$$(1-\mathcal{P}) > \sum_{i=6t}^{9t} \frac{3^{24t-i}}{4^{24t}} \binom{24t}{i} > 5(2^t) \sum_{i=6t}^{9t} \frac{3^{21t-i}}{4^{24t}} \binom{24t}{i+3t} > (2^t)\mathcal{P}.$$

This shows $\mathcal{P} < 1/2^t$ which completes the proof of lemma 3.

Reducing Logarithms in Totally Non-maximal Imaginary Quadratic Orders to Logarithms in Finite Fields

Detlef Hühnlein¹ and Tsuyoshi Takagi²

¹ Security Networks GmbH Mergenthalerallee 77-81, D-65760 Eschborn, Germany huehnlein@secunet.de
² NTT Information Sharing Platform Laboratories Immermannstr. 40, D-40210 Düsseldorf, Germany ttakagi@ntt.de

Abstract. We discuss the discrete logarithm problem over the class group $Cl(\Delta)$ of an imaginary quadratic order \mathcal{O}_{Δ} , which was proposed as a public-key cryptosystem by Buchmann and Williams [8]. While in the meantime there has been found a subexponential algorithm for the computation of discrete logarithms in $Cl(\Delta)$ [16], this algorithm only has running time $L_{\Delta}[\frac{1}{2},c]$ and is far less efficient than the number field sieve with $L_p[\frac{1}{3},c]$ to compute logarithms in \mathbb{F}_p^* . Thus one can choose smaller parameters to obtain the same level of security. It is an open question whether there is an $L_{\Delta}[\frac{1}{3},c]$ algorithm to compute discrete logarithms in arbitrary $Cl(\Delta)$.

In this work we focus on the special case of totally non-maximal imaginary quadratic orders \mathcal{O}_{Δ_p} such that $\Delta_p = \Delta_1 p^2$ and the class number of the maximal order $h(\Delta_1) = 1$, and we will show that there is an $L_{\Delta_p}[\frac{1}{3},c]$ algorithm to compute discrete logarithms over the class group $Cl(\Delta_p)$. The logarithm problem in $Cl(\Delta_p)$ can be reduced in (expected) $O(\log^3 p)$ bit operations to the logarithm problem in \mathbb{F}_p^* (if $(\frac{\Delta_1}{p}) = 1$) or $\mathbb{F}_{p^2}^*$ (if $(\frac{\Delta_1}{p}) = -1$) respectively. This result implies that the recently proposed efficient DSA-analogue in totally non-maximal imaginary quadratic order \mathcal{O}_{Δ_p} [21] are only as secure as the original DSA scheme based on finite fields and hence loose much of its attractiveness.

1 Introduction

A general and possible inherent problem of all currently known public key cryptosystems is that their intractability is based on certain unproven assumptions. Thus nobody can guarantee that popular cryptosystems based on factoring integers or computing discrete logarithms in some group will remain secure in the future. Therefore it is important to study alternative primitives and different groups to have a backup if one assumption such as the intractability of factoring or computation of discrete logarithms in one group turns out to be false. Beside the multiplicative group of finite fields and the group of points on (hyper-) elliptic curves over finite fields, a very promising candidate for a group in which the discrete logarithm is hard is the class group $Cl(\Delta)$ of imaginary quadratic orders, such as proposed by Buchmann and Williams [8] in 1988. For example the discrete logarithm problem in $Cl(\Delta)$ has the interesting property that it is at least as hard as factoring the discriminant Δ . Another reason which makes studying imaginary quadratic orders \mathcal{O}_{Δ} very important today, is that these rings are isomorphic to the endomorphism rings of non-supersingular elliptic curves over finite fields. Thus a good understanding of these rings can shed some light on the real difficulty of the discrete logarithm problem in elliptic curves. While Hafner and McCurley discovered a subexponential algorithm one year later [16] to compute discrete logarithms in $Cl(\Delta)$, this algorithm has a running time $L_{\Delta}[\frac{1}{2},1]$ and is far less efficient than the number field sieve to compute discrete logarithms in \mathbb{F}_p^* or factoring integers with $L_n[\frac{1}{3},(\frac{64}{9})^{1/3}]$. The precise definition of $L_n[e,c]$ will be given in Section 3. Thus one may choose smaller parameters, and still obtain the same level of security. It is an open question whether there is an $L_{\Delta}[\frac{1}{3},c]$ algorithm to compute discrete logarithms in arbitrary imaginary quadratic class groups $Cl(\Delta)$. Note that as mentioned above this would imply another asymptotically fast algorithm for factoring integers, because factoring the discriminant Δ is reduced to the computation of discrete logarithms in $Cl(\Delta)$.

Furthermore these cryptosystems based on imaginary quadratic class groups are not only interesting from a theoretical point of view. Recently cryptosystems have been proposed with very practical properties. We will only name a few cryptosystems based on imaginary quadratic orders here and refer to Section 2 for a more comprehensive survey. In [26] a public key cryptosystem was proposed with quadratic decryption time. To our knowledge this is the only known crypto system having this property. First implementations show that the decryption is as efficient as RSA-encryption with $e = 2^{16} + 1$. While this cryptosystem is based on factoring, it is also possible to set up interesting DL-based cryptosystems using non-maximal imaginary quadratic orders. If one uses the recently developed exponentiation technique for totally non-maximal orders [21] it is possible to implement efficient DSA-analogues. The running time is roughly comparable to DSA in \mathbb{F}_p^* and there is certainly much space for further improvements. The major property of these totally non-maximal orders is that the class number of the maximal order $h(\Delta_1) = 1$ and thus the class number of the non-maximal order $h(\Delta_p) = p - (\frac{\Delta_1}{p})$, where the conductor p is prime and $(\frac{\Delta_1}{p})$ is the Kronecker-Symbol, is known immediately. Note that these totally non-maximal quadratic orders are therefore analogous to supersingular elliptic curves, where one also knows the group order in advance.

In this work we will show that the discrete logarithm problem in totally non-maximal imaginary quadratic orders can be reduced to the discrete logarithm problem in \mathbb{F}_p^* (if $(\frac{\Delta_1}{p}) = 1$) or $\mathbb{F}_{p^2}^*$ (if $(\frac{\Delta_1}{p}) = -1$) respectively. The reduction is very efficient and can be performed in (expected) $O(\log^3 p)$ bit operations. Thus the situation for cryptosystems based on imaginary quadratic orders is

somewhat analogous to the situation for cryptosystems based on elliptic curves. This may be summarized as follows:

While there is no known algorithm with $L_{\Delta}[\frac{1}{3}, c]$ for the computation of discrete logarithms in imaginary quadratic class groups in general, there are problem classes for which such an algorithm is known. This is no general problem however, because it is easy to avoid these weak classes in practice.

It is clear that an analogous statement for elliptic curves would be somewhat sharper and consider algorithms with subexponential running time $L_p[e,c]$, e < 1.

This paper is organized as follows: In Section 2 we will give a brief survey of cryptosystems based on imaginary quadratic orders, because many results appeared very recently and are sometimes not yet published. Section 3 gives the necessary background and notations of imaginary quadratic orders. In Section 4 we will provide the main result of this paper which consists of the reduction of the discrete logarithm problem in totally non-maximal imaginary quadratic orders to the discrete logarithm problem in finite fields. Finally, in Section 5, we will conclude this work by discussing the cryptographic implications of our result.

2 A Brief Survey of Cryptosystems Based on Imaginary Quadratic Orders

We will only highlight the most important works in this direction. As mentioned above it is a general problem that the security of popular cryptosystems is based on *unproven assumptions*. Nobody can guarantee that DL-type cryptosystems based on finite fields or elliptic curves over finite fields will stay secure forever. Thus it is important to study alternative groups which can be used if an efficient algorithm for the computation of discrete logarithms in one particular type of group is discovered.

2.1 The Early Days - Maximal Orders

With this motivation Buchmann and Williams [8] proposed to use imaginary quadratic class groups $Cl(\Delta)$ for the construction of cryptosystems. A nice property of this approach is that breaking this scheme is at least as difficult as factoring the fundamental discriminant Δ of the maximal order. Furthermore it should be mentioned that imaginary quadratic orders are closely related to non-supersingular elliptic curves over finite fields. They happen to be isomorphic to their endomorphism ring. Thus a sound understanding of imaginary quadratic orders may lead to a better understanding of the real security of elliptic curve cryptosystems. In 1988, when they proposed these groups for cryptographic purposes, the best algorithms to compute the class number $h(\Delta)$ and discrete logarithms in $Cl(\Delta)$ were exponential time algorithms with $L_{\Delta}[1, \frac{1}{5}]$ [23,30] assuming

the truth of the Generalized Riemann Hypothesis (GRH) or $L_{\Delta}[1, \frac{1}{4}]$ without this assumption. In [6] the first implementation was reported along with a complexity analysis of this key agreement scheme. For example it was shown that the complexity of an exponentiation in $Cl(\Delta)$ needs $O(\log^4 |\Delta|)$ bit operations, which is fairly inefficient compared to the original scheme [13] which is of cubic complexity. Another problem of cryptosystems based on class group $Cl(\Delta)$ of the maximal order, was that the computation of the class number $h(\Delta)$ is almost as difficult as the computation of discrete logarithms. Thus it seemed impossible to set up signature schemes analogous to DSA [25] or RSA [28].

Even worse for this approach was the discovery of a subexponential time algorithm [16] by Hafner and McCurley in 1989. This algorithm has running time $L_{\Delta}[\frac{1}{2},c]$ and can be used to compute the class number $h(\Delta)$ and with some modifications to the computation of discrete logarithms in $Cl(\Delta)$ as shown in [5]. Note that at this time the asymptotically best algorithm for factoring integers was the quadratic sieve [29] with running time $L_n[\frac{1}{2},1]$ if one makes certain plausible assumptions. The situation for discrete logarithms in \mathbb{F}_p^* was similar these days. The algorithm due to Coppersmith, Odlyzko and Schroeppel (COS) [11] to compute discrete logarithms in prime fields also has running time $L_p[\frac{1}{2},1]$.

Thus, it was inclined to consider cryptosystems based on imaginary quadratic class groups $Cl(\Delta)$ to be unsuitable for practical application.

2.2 The Recent Revival - Non-maximal Orders

In the meantime however an idea of Pollard lead to today's asymptotically best algorithm for factoring integers - the number field sieve (see [24]). This algorithm has (expected) running time $L_n[\frac{1}{3},(\frac{64}{9})^{1/3}]$ and was used in 1996 for the factorization of RSA-130 [9] and recently for the factorization of RSA-140 [27] for example. The number field sieve can also be used to compute discrete logarithms in finite fields (see e.g. [15,31]), where the (expected) running time is $L_p[\frac{1}{3},(\frac{64}{9})^{1/3}]$ as well. In contrast to this development there is still no $L_\Delta[\frac{1}{3},c]$ algorithm known for the computation of discrete logarithms in arbitrary $Cl(\Delta)$. The asymptotically best algorithm for this task still is an analogue of the multiple polynomial quadratic sieve [22] with $L_\Delta[\frac{1}{2},1]$.

It is clear that this development alone would not justify the term "revival". In 1998 it was shown in [19] that by using class groups $Cl(\Delta_p)$, $\Delta_p = \Delta_1 p^2$, of non-maximal orders one solves the problem that the class number $h(\Delta_p)$ can not be determined and that one is able to implement an ElGamal-type cryptosystem with comparably fast decryption. While the performance of this scheme still was too bad to be used in practice this result may be considered as the birth of a new generation of cryptosystems based on quadratic orders.

Recently, a very efficient successor [26] with quadratic decryption time was proposed. This scheme was later on called NICE for New Ideal Coset Encryption. First implementations show that the time for decryption is comparable to RSA - encryption with $e = 2^{16} + 1$. The central idea is to use an element $\mathfrak{g} \in \ker(\varphi^{-1})$ to mask the message in the ElGamal-type encryption scheme by multiplication

with \mathfrak{g}^k for random k. Here φ^{-1} is the isomorphism introduced in [19] which allows switching from the public non-maximal order to the secret maximal order. Thus during the decryption step, which essentially consists of the computation of φ^{-1} , the mask \mathfrak{g}^k simply disappears and the message is recovered. Note that the computation of φ^{-1} is essentially one modular inversion with the Extended Euclidean Algorithm which takes quadratic time. It is clear that this cryptosystem is very well suited for applications in which a central server has to decrypt a large amount of ciphertext in a short time. For this scenario one may use the recently developed NICE-batch-decryption method [21], which speeds up the already very efficient decryption process by another 30% for a batch size of 100 messages. An efficient undeniable signature scheme based on the NICE-structure was also proposed [3].

In 1998 the first conventional signature schemes based on non-maximal imaginary quadratic orders were also proposed. In [20] RSA- and Rabin analogues were proposed. The corresponding encryption schemes have the major advantage that they are immune against low-exponent- and chosen-ciphertext attacks. Moreover a novel algorithm to compute square roots in $Cl(\Delta_p)$ was proposed, which replaces the fairly inefficient Gaussian algorithm using ternary quadratic forms. To avoid the computation of $h(\Delta_1)$, where $|\Delta_1|$ should have at least 200 bits to prevent the factorization of Δ_p using ECM (see [4] for a recent finding of a 53 digit factor), it was proposed to use totally non-maximal imaginary quadratic orders. Note that the above cryptosystems are based on completely factoring the non-fundamental discriminant Δ_p or Δ_{pq} in the case of totally non-maximal orders respectively. While the utilization of totally non-maximal orders for RSA-analogues is only interesting from a theoretical point of view, it is clear that this structure may well be used to set up DSA analogues. The discriminant $\Delta_p = \Delta_1 p^2$, with $\Delta_1 = -163$ and hence $h(\Delta) = 1$ for example, can be chosen with about 800 bits to obtain the same level of security as for DSA in \mathbb{F}_p^* with p about 1000 bits. Note that this comparison, i.e. 400 bit p for $Cl(\Delta_p)$ compared to 1000 bit p for \mathbb{F}_p^* , is a rather pessimistic one. Nevertheless this DSA analogue *seemed* to be too inefficient to be used in practice.

Very recently however a new arithmetic for these totally non-maximal orders was proposed [21]. The central idea is to replace the fairly inefficient conventional ideal-arithmetic, i.e. multiplication and reduction of ideals, by simple manipulations on the corresponding generator in the maximal order. This means that instead of (multiple) applications of the comparably costly Extended Euclidean Algorithm one only has a few modular multiplications. This strategy turns out to be thirteen times as fast and ends up with a DSA analogue based on totally non-maximal orders, in which the running time for the signature generation is roughly comparable to the conventional DSA in \mathbb{F}_p^* . Furthermore there still seems to be much space for further improving this scheme.

However beside the possibility to *speed up* the DSA analogues, there is yet another and even more important effect of the very recent result [21]:

It was precisely the way in which one considers the arithmetic of ideals in totally non-maximal orders, which led to the (previously conjectured)

constructive version of the reduction proof presented in Section 4 of this work.

3 Some Background and Notations Concerning Imaginary Quadratic Orders

We first define the function $L_n[e,c]$ which is used to describe the asymptotic running time of subexponential algorithms. Let $n,e,c\in\mathbb{R}$ with $0\leq e\leq 1$ and c>0. Then we define

$$L_n[e,c] = \exp\left(c \cdot (\log|n|)^e \cdot (\log\log|n|)^{1-e}\right).$$

Thus the running time for subexponential algorithms is between polynomial time $(L_n[0,c])$ and exponential time $(L_n[1,c])$.

Now we will give some basics concerning quadratic orders. The basic notions of imaginary quadratic number fields may be found in [7,10]. For a more comprehensive treatment of the relationship between maximal and non-maximal orders we refer to [12,19].

Let $\Delta \equiv 0, 1 \mod 4$ be a negative integer, which is not a square. The quadratic order of discriminant Δ is defined to be

$$\mathcal{O}_{\Lambda} = \mathbb{Z} + \omega \mathbb{Z}$$

where

$$\omega = \begin{cases} \sqrt{\frac{\Delta}{4}}, & \text{if } \Delta \equiv 0 \pmod{4}, \\ \frac{1+\sqrt{\Delta}}{2}, & \text{if } \Delta \equiv 1 \pmod{4}. \end{cases}$$
 (1)

The standard representation of some $\alpha \in \mathcal{O}_{\Delta}$ is $\alpha = x + y\omega$, where $x, y \in \mathbb{Z}$.

If Δ_1 is squarefree, then \mathcal{O}_{Δ_1} is the maximal order of the quadratic number field $\mathbb{Q}(\sqrt{\Delta_1})$ and Δ_1 is called a fundamental discriminant. The non-maximal order of conductor p > 1 with (non-fundamental) discriminant $\Delta_p = \Delta_1 p^2$ is denoted by \mathcal{O}_{Δ_p} . We will always assume in this work that the conductor p is prime. Furthermore we will omit the subscripts to reference arbitrary (fundamental or non-fundamental) discriminants. Because $\mathbb{Q}(\sqrt{\Delta_1}) = \mathbb{Q}(\sqrt{\Delta_p})$ we also omit the subscripts to reference the number field $\mathbb{Q}(\sqrt{\Delta})$. The standard representation of an \mathcal{O}_{Δ} -ideal is

$$\mathfrak{a} = q\left(aZ + \frac{b + \sqrt{\Delta}}{2}Z\right) = q(a, b), \tag{2}$$

where $q \in \mathbb{Q}_{>0}$, $a \in \mathbb{Z}_{>0}$, $c = (b^2 - \Delta)/(4a) \in \mathbb{Z}$, $\gcd(a,b,c) = 1$ and $-a < b \le a$. The norm of this ideal is $\mathcal{N}(\mathfrak{a}) = aq^2$. An ideal is called primitive if q = 1. A primitive ideal is called reduced if $|b| \le a \le c$ and $b \ge 0$, if a = c or |b| = a. It can be shown, that the norm of a reduced ideal \mathfrak{a} satisfies $\mathcal{N}(\mathfrak{a}) \le \sqrt{|\Delta|/3}$ and conversely that if $\mathcal{N}(\mathfrak{a}) \le \sqrt{|\Delta|/4}$ then the primitive ideal \mathfrak{a} is reduced. We denote the reduction operator in the maximal order by $\rho_1()$ and write $\rho_p()$ for the reduction operator in the non-maximal order of conductor p.

The group of invertible \mathcal{O}_{Δ} -ideals is denoted by \mathcal{I}_{Δ} . Two ideals $\mathfrak{a}, \mathfrak{b}$ are equivalent, if there is a $\gamma \in \mathbb{Q}(\sqrt{\Delta})$, such that $\mathfrak{a} = \gamma \mathfrak{b}$. This equivalence relation is denoted by $\mathfrak{a} \sim \mathfrak{b}$. The set of principal \mathcal{O}_{Δ} -ideals, i.e. which are equivalent to \mathcal{O}_{Δ} , is denoted by \mathcal{P}_{Δ} . The factor group $\mathcal{I}_{\Delta}/\mathcal{P}_{\Delta}$ is called the class group of \mathcal{O}_{Δ} denoted by $Cl(\Delta)$. $Cl(\Delta)$ is a finite abelian group with neutral element \mathcal{O}_{Δ} . In every equivalence class there is one and only one reduced ideal, which represents its class. Algorithms for the group operation (multiplication and reduction of ideals) can be found in [10]. The order of the class group is called the class number of \mathcal{O}_{Δ} and is denoted by $h(\Delta)$.

All cryptosystems from Section 2.2 make use of the relation between the maximal and some non-maximal order. Any non-maximal order of conductor p may be represented as $\mathcal{O}_{\Delta_p} = \mathbb{Z} + p\mathcal{O}_{\Delta_1}$. A special type of non-maximal order, which is of central importance in this work, is given if $h(\Delta) = 1$. In this case \mathcal{O}_{Δ_p} is called a *totally non-maximal* imaginary quadratic order. An \mathcal{O}_{Δ} -ideal \mathfrak{a} is called prime to p, if $\gcd(\mathcal{N}(\mathfrak{a}), p) = 1$. It is well known, that all \mathcal{O}_{Δ_p} -ideals prime to the conductor are invertible.

Denote by $\mathcal{I}_{\Delta_p}(p)$ (respectively, $\mathcal{P}_{\Delta_p}(p)$) the \mathcal{O}_{Δ_p} -ideals prime to p (respectively, the principal \mathcal{O}_{Δ_p} -ideals prime to p). There is an isomorphism (See [12, Proposition 7.22,page 145])

$$\mathcal{I}_{\Delta_p}(p) / \mathcal{P}_{\Delta_p}(p) \simeq \mathcal{I}_{\Delta_p} / \mathcal{P}_{\Delta_p} = Cl(\Delta_p).$$
 (3)

Thus we may 'neglect' the ideals which are not prime to the conductor, if we are only interested in the class group $Cl(\Delta_p)$. There is an isomorphism between the group of \mathcal{O}_{Δ_p} -ideals which are prime to p and the group of \mathcal{O}_{Δ_1} -ideals, which are prime to p, denoted by $\mathcal{I}_{\Delta_1}(p)$ respectively:

Proposition 1. Let \mathcal{O}_{Δ_p} be an order of conductor p in an imaginary quadratic field $\mathbb{Q}(\sqrt{\Delta})$ with maximal order \mathcal{O}_{Δ_1} .

- (i.) If $\mathfrak{A} \in \mathcal{I}_{\Delta_1}(p)$, then $\mathfrak{a} = \mathfrak{A} \cap \mathcal{O}_{\Delta_p} \in \mathcal{I}_{\Delta_p}(p)$ and $\mathcal{N}(\mathfrak{A}) = \mathcal{N}(\mathfrak{a})$.
- (ii.) If $\mathfrak{a} \in \mathcal{I}_{\Delta_p}(p)$, then $\mathfrak{A} = \mathfrak{a}\mathcal{O}_{\Delta_1} \in \mathcal{I}_{\Delta_1}(p)$ and $\mathcal{N}(\mathfrak{a}) = \mathcal{N}(\mathfrak{A})$.
- (iii.) The map $\varphi: \mathfrak{A} \mapsto \mathfrak{A} \cap \mathcal{O}_{\Delta_p}$ induces an isomorphism $\mathcal{I}_{\Delta_1}(p) \xrightarrow{\sim} \mathcal{I}_{\Delta_p}(p)$. The inverse of this map is $\varphi^{-1}: \mathfrak{a} \mapsto \mathfrak{a} \mathcal{O}_{\Delta_1}$.

Proof. See [12, Proposition 7.20, page 144].

Thus we are able to switch to and from the maximal order as applied in the cryptosystems of Section 2.2. The algorithms $\mathsf{GoToMaxOrder}(\mathfrak{a},p)$ to compute φ^{-1} and $\mathsf{GoToNonMaxOrder}(\mathfrak{A},p)$ to compute φ respectively may be found in [19]. Note, that the above map is defined on ideals themselves, rather than equivalence classes. The class group $Cl(\Delta_p)$ of a non-maximal order can be described as follows:

Proposition 2. There is an isomorphism

$$Cl(\Delta_p) \simeq \mathcal{I}_{\Delta_1}(p) / \mathcal{P}_{\Delta_1, \mathbb{Z}}(p),$$

where $\mathcal{P}_{\Delta_1,\mathbb{Z}}(p)$ denotes the subgroup of $\mathcal{I}_{\Delta_1}(p)$ generated by the principal ideals of the form $\alpha \mathcal{O}_{\Delta_1}$ where $\alpha \in \mathcal{O}_{\Delta_1}$ satisfies $\alpha \equiv a \mod p \mathcal{O}_{\Delta_1}$ for some $a \in \mathbb{Z}$ such that $\gcd(a,p)=1$. This isomorphism is induced by isomorphism φ between $\mathcal{I}_{\Delta_1}(p)$ and $\mathcal{I}_{\Delta_p}(p)$.

Proof. See the details in [12, Proposition 7.22,page 145]. For the sake of convenience we describe the outline of proof. Recall $\mathcal{I}_{\Delta_p}(p)/\mathcal{P}_{\Delta_p}(p) \simeq Cl(\Delta_p)$. Isomorphism $\varphi^{-1}: \mathfrak{a} \mapsto \mathfrak{a}\mathcal{O}_{\Delta_1}$ maps $\mathcal{P}_{\Delta_p}(p)$ to a subgroup \mathcal{P}' of $\mathcal{I}_{\Delta_1}(p)$. We can prove that $\mathcal{P}' = \mathcal{P}_{\Delta_1} \mathbb{Z}(p)$ using relation

$$\alpha \equiv a \bmod p\mathcal{O}_{\Delta_1}, a \in \mathbb{Z}, \gcd(a,p) = 1 \Longleftrightarrow \alpha \in \mathcal{O}_{\Delta_p}, \gcd(N(\alpha),p) = 1.$$

This interpretation of $Cl(\Delta_p)$ will be used in Section 4 to reduce the computation of discrete logarithms in totally non-maximal imaginary quadratic orders to the computation of discrete logarithms in finite fields.

Definition 1. Let $\Delta_1 < 0$ and $\Delta_1 \equiv 0, 1 \mod 4$, such that $h(\Delta_1) = 1$ and p prime. Furthermore let \mathfrak{g} and \mathfrak{a} be reduced \mathcal{O}_{Δ_p} -ideals in standard-representation (2), which represent classes of the class group $Cl(\Delta_p)$ of the totally non-maximal order. Then the discrete logarithm problem DLP in $Cl(\Delta_p)$ is given as follows: Determine an $a \in \mathbb{Z}$ such that $\mathfrak{g}^a \sim \mathfrak{a}$, or show that no such a exists.

Furthermore the class number of a totally non-maximal order of conductor p is given as follows:

Proposition 3. Let $\Delta_1 < -4$, $\Delta_1 \equiv 0, 1 \mod 4$ such that $h(\Delta_1) = 1$ and p prime. Then $h(\Delta_p) = p - (\frac{\Delta_1}{p})$, where $(\frac{\Delta_1}{p})$ is the Kronecker-symbol.

Proof. This follows immediately from [12, Theorem 7.24, page 146]. \Box

Finally we will make use of the following interpretation of the ring $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$:

Proposition 4. Let \mathcal{O}_{Δ_1} be the maximal order and p be the prime conductor. Then there is an isomorphism between

$$(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \simeq \mathbb{F}_p[X]/(f(X)),$$

where (f(X)) is the ideal generated by $f(X) \in \mathbb{F}_p[X]$ and

$$f(X) = \begin{cases} X^2 - \frac{\Delta_1}{4}, & \text{if } \Delta \equiv 0 \pmod{4}, \\ X^2 - X + \frac{1 - \Delta_1}{4}, & \text{if } \Delta \equiv 1 \pmod{4}. \end{cases}$$
 (4)

Proof. Let $\rho \in \mathbb{F}_p$ be a root of $f(X) \in \mathbb{F}_p[X]$, where f(X) is as given above. Then an element $\alpha \in \mathbb{F}_p[X]/(f(X))$ has a representation $\alpha = x + y\rho$, where $x, y \in \mathbb{F}_p$. On the other hand an element $\beta \in (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ has a representation $\beta = \bar{x} + \bar{y}\omega$, where ω given as in (1) and $\bar{x}, \bar{y} \in \mathbb{F}_p$. If we set $x \equiv \bar{x} \mod p$ and $y \equiv \bar{y} \mod p$ we immediately have the desired bijective correspondence. Furthermore it can be easily shown by straight forward calculation that this bijective correspondence is indeed an isomorphism.

Note that this isomorphism implicitly was used in [21] to speed up the arithmetic in totally-non-maximal orders.

4 Reducing Logarithms in Totally Non-maximal Orders to Logarithms in Finite Fields

In this section we will show that the discrete logarithm problem in $Cl(\Delta_p)$ as given in Definition 1 can be reduced to the discrete logarithm problem in finite fields. More precisely we will show the following

Theorem 1. The DLP in the class group $Cl(\Delta_p)$ of a totally non-maximal order \mathcal{O}_{Δ_p} , where $\Delta_p = \Delta_1 p^2$ for prime p, can be reduced in (expected) $O(\log^3 p)$ bit operations

- 1. to the DLP in $\mathbb{F}_{p^2}^*$ if $(\frac{\Delta_1}{p}) = -1$ or
- 2. to the DLP in \mathbb{F}_p^* if $(\frac{\Delta_1}{p}) = 1$.

To show the above result we will first consider the structure of the class group $Cl(\Delta_p)$ of the totally non-maximal order. By the definition of a totally non-maximal order, we know that the class number of the maximal order $h(\Delta_1) = 1$. This means that in \mathcal{O}_{Δ_1} there are only principal ideals and hence $\mathcal{I}_{\Delta_1} = \mathcal{P}_{\Delta_1}$. Recall from Proposition 2 that $Cl(\Delta_p) \simeq \mathcal{I}_{\Delta_1}(p)/\mathcal{P}_{\Delta_1,\mathbb{Z}}(p)$, where $\mathcal{P}_{\Delta_1,\mathbb{Z}}(p)$ denotes the principal ideals $\alpha\mathcal{O}_{\Delta_1}$ of the form $\alpha \equiv a \mod p\mathcal{O}_{\Delta_1}$, with $a \in \mathbb{Z}$ and $\gcd(a,p) = 1$. Thus in our case we obtain the following isomorphism:

$$Cl(\Delta_p) \simeq \mathcal{P}_{\Delta_1}(p) / \mathcal{P}_{\Delta_1, Z\!Z}(p)$$

Hence the group structure of the class group $Cl(\Delta_p)$ can be explained exclusively by a relation of principal ideals in the maximal order \mathcal{O}_{Δ_1} . With this knowledge we are able to relate the ring $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ to our class group $Cl(\Delta_p)$.

Lemma 1. The map $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \to \mathcal{P}_{\Delta_1}(p)/\mathcal{P}_{\Delta_1,\mathbb{Z}}(p)$, which $\alpha \in (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ maps to $\alpha\mathcal{O}_{\Delta_1} \in \mathcal{P}_{\Delta_1}(p)/\mathcal{P}_{\Delta_1,\mathbb{Z}}(p)$, is a well-defined group homomorphism and surjective.

Proof. This is shown in the more comprehensive proof of Theorem 7.24 in [12] (page 147). \Box

The "running time" to compute this map is trivially constant time. Note that this map cannot be injective, just because there are (depending on (Δ_1/p)) either p^2-1 or $(p-1)^2$ elements in $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ and by Proposition 3 only $p-(\Delta_1/p)=p\pm 1$ elements in $Cl(\Delta_p)$. It would be an isomorphism if we would restrict it to appropriate subgroups of $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$. The precise relation is given in Lemma 2.

In the next step we show that there is an isomorphism ψ between the ring $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ and the multiplicative group of a finite field of degree at most 2, which can be computed in (expected) $O(\log^3 p)$ bit operations.

Lemma 2. We have to distinguish two cases:

- 1. If $(\frac{\Delta_1}{p}) = -1$ then there exists an isomorphism $\psi : (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \to \mathbb{F}_{p^2}^*$, which can be computed in constant time.
- 2. If $(\frac{\Delta_1}{p}) = 1$ then there exits a surjective homomorphism $\psi : (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \to \mathbb{F}_p^*$, which can be computed with (expected) $O(\log^3 p)$ bit operations.

Proof. From Proposition 4 we know that there is an isomorphism $(\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \to \mathbb{F}_p[X]/(f(X))$, where $f(X) \in \mathbb{F}_p[X]$ is given as in (4). Now we need to separate the two cases.

- (1) $(\frac{\Delta_1}{p}) = -1$: In this case the polynomial f(X) is irreducible in $\mathbb{F}_p[X]$ and therefore we have $\mathbb{F}_p[X]/(f(X)) \simeq \mathbb{F}_{p^2}$. Therefore we get the bijective map $\psi: (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^* \to \mathbb{F}_{p^2}^*$ as follows: Let $\alpha = a + b\omega \in (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$. Then $\psi(\alpha) = a + bX \in \mathbb{F}_{p^2}$. This map can be trivially computed in constant time. Furthermore it is easy to show that this map is indeed an isomorphism.
- (2) $\left(\frac{\Delta_1}{p}\right) = 1$: In this case the polynomial f(X) is not irreducible, but can be decomposed as $f(X) = (X \rho)(X \bar{\rho}) \in \mathbb{F}_p[X]$ where $\rho \in \mathbb{F}_p$ is a root of f(X) and $\bar{\rho}$ is conjugate to ρ . Thus if $\Delta_1 \equiv 0 \mod 4$ and $D = \Delta_1/4$ we have $\rho \in \mathbb{F}_p$ such that $\rho^2 \equiv D \mod p$ and $\bar{\rho} = -\rho$. In the other case $\Delta_1 \equiv 1 \mod 4$ we have $\rho = (1+b)/2$, where $b^2 \equiv \Delta_1 \mod p$ and $\bar{\rho} = (1-b)/2 \in \mathbb{F}_p$. Thus in our case $(\Delta_1/p) = 1$ we have $\mathbb{F}_p[X]/(f(X)) \simeq \mathbb{F}_p[X]/(X \rho) \otimes \mathbb{F}_p[X]/(X \bar{\rho})$. In both cases $(\Delta_1 \text{ even or odd})$ we have to compute a square root in \mathbb{F}_p to find ρ and $\bar{\rho}$. This takes random polynomial time using the algorithm of Cipolla. More precisely we know from [1, Theorem 7.2.3, page 158] that this algorithm takes (expected) time $O(\log^3 p)$. In this case we have the map between $\alpha = a + b\omega \in (\mathcal{O}_{\Delta_1}/p\mathcal{O}_{\Delta_1})^*$ and $\psi(\alpha) = a + b\rho \in \mathbb{F}_p^*$. Finally one can easily show that this map is indeed a surjective homomorphism.

Now we only have one more minor problem. The DLP in Definition 1 is formulated for reduced ideals in the standard representation such that $\mathfrak{a} = aZ + \frac{b+\sqrt{\Delta_1}}{2}Z$ in $Cl(\Delta_p)$. We have to convert this standard representation in $Cl(\Delta_p)$ to that in $\mathcal{P}_{\Delta_1}(p)/\mathcal{P}_{\Delta_1,ZZ}(p)$ using Proposition 2. The following simple lemma indicates that we can efficiently switch to the desired generator-representation (and back).

Lemma 3. Let $\Delta_1 < 0$ and $\Delta_1 \equiv 0, 1 \mod 4$ such that $h(\Delta_1) = 1$ and p prime. Then

- 1. there is a deterministic algorithm which computes ideal $\alpha \mathcal{O}_{\Delta_1} = \varphi^{-1}(\mathfrak{a}) \in \mathcal{P}_{\Delta_1}(p)$ for a given reduced ideal $\mathfrak{a} \in Cl(\Delta_p)$ prime to p in $O(\log^2 p)$ bit operations and
- 2. there is a deterministic algorithm which computes reduced ideal \mathfrak{a} which is equivalent to $\varphi(\alpha \mathcal{O}_{\Delta_1}) \in Cl(\Delta_p)$ for a given ideal $\alpha \mathcal{O}_{\Delta_1} \in \mathcal{P}_{\Delta_1}(p)/\mathcal{P}_{\Delta_1,\mathbb{Z}}(p)$ in $O(\log^2 p)$ bit operations.

Proof. Note that algorithm φ and φ^{-1} can be computed in $O(\log^2 p)$ bit operations [26]. We denote by $\mathfrak{a} = aZ + \frac{b + \sqrt{\Delta_q}}{2}Z$ a reduced ideal in $Cl(\Delta_p)$.

From Proposition 1 all reduced ideals $\mathfrak{a} \in Cl(\Delta_p)$ prime to p are of the form $\mathfrak{a} = \varphi(\alpha \mathcal{O}_{\Delta_1})$ for some $\alpha \in \mathcal{O}_{\Delta_1}$. We can find the generator α by reducing $\varphi^{-1}(\mathfrak{a})$. Let $\mathfrak{A} = \varphi^{-1}(\mathfrak{a}) = A\mathbb{Z} + \frac{B+\sqrt{\Delta_1}}{2}\mathbb{Z}$. From [2] one can reduce ideal \mathfrak{A} of \mathcal{O}_{Δ_1} and find element $\alpha \in \mathcal{O}_{\Delta_1}$ such that $\alpha \mathcal{O}_{\Delta_1} \sim \mathfrak{A}$ in $O((\log A)^2)$ bit operations. The norm of ideal $\mathfrak{a} \in Cl(\Delta_p)$ is a. Because \mathfrak{a} is reduced we have $a < \sqrt{|\Delta_p|/3}$ and a = O(p). Note that the norm a of ideals does not change while switching the orders by map φ , thus A = a holds. Therefore one can compute the generator $\alpha \mathcal{O}_{\Delta_1} = \varphi^{-1}(\mathfrak{a})$ in $O(\log^2 p)$ bit operations. On the contrary, let \mathfrak{A} be the standard representation of ideal $\alpha \mathcal{O}_{\Delta_1} \in \mathcal{P}_{\Delta_1}(p)/\mathcal{P}_{\Delta_1,\mathbb{Z}}(p)$. From [21] one can compute ideal \mathfrak{A} in $O((\log \Delta_1)^2)$ bit operations. Then to compute the reduced ideal equivalent to $\varphi(\mathfrak{A})$ in $Cl(\Delta_p)$ requires map φ and one reduction algorithm, and they are in $O(\log^2 p)$ bit operations. This proofs the assertion of Lemma 3.

Thus we are now able to put together our auxiliary lemma to prove the main result of this work.

Proof (Proof of Theorem 1). If one is given \mathfrak{g} , \mathfrak{a} as given in Definition 1 to compute the discrete logarithm in the class group $Cl(\Delta_p)$ then one can compute the corresponding generators $\gamma, \alpha \in \mathcal{O}_{\Delta_1}$ such that $\gamma \mathcal{O}_{\Delta_1} = \varphi^{-1}(\mathfrak{g}), \alpha \mathcal{O}_{\Delta_1} = \varphi^{-1}(\mathfrak{g})$ by Lemma 1 and Lemma 3. Using the isomorphism ψ from Lemma 2 one can compute the corresponding elements $a = \psi(\alpha)$ and $g = \psi(\gamma)$ in the finite field \mathbb{F}_p^* (if $(\Delta_1/p) = 1$) or $\mathbb{F}_{p^2}^*$ (if $(\Delta_1/p) = -1$) respectively. Then one is able to compute the discrete logarithm there or determine that it does not exist. It is clear that the entire reduction does only take (expected) $O(\log^3 p)$ bit operations.

5 Conclusion

In this work we have shown that the discrete logarithm problem in the class group $Cl(\Delta_p)$ of a totally non-maximal imaginary quadratic order can be reduced to the discrete logarithm problem in finite fields using (expected) $O(\log^3 p)$ bitoperations. This result clearly implies that the formerly proposed bitlength of 800 for Δ_p does not provide sufficient security, because one could simply compute discrete logarithms in $\mathbb{F}_{p^k}^*$, where $k \in \{1,2\}$ which should be possible in the near future if $p \approx 2^{400}$. The algorithm which is used in $\mathbb{F}_{p^k}^*$ is the number field sieve with $L[\frac{1}{3}]$. This would imply that p (at least in the case that $(\Delta_1/p) = 1$) should be about 1024 bit to yield (expected) long term security. Hence cryptosystems based on totally non-maximal imaginary quadratic orders seem to lose much of their attractiveness.

Analogous to the situation for Elliptic Curves, where the DLP in supersingular curves can efficiently be solved in finite fields with small extension degree, we discovered that there is also a weak class for class groups of imaginary quadratic orders. It remains an open question whether it is possible to find an $L[\frac{1}{3}]$ algorithm to compute discrete logarithms in arbitrary class groups. Another

interesting question is whether these results have any relevance to the elliptic curves discrete logarithm problem for elliptic curves whose endomorphism ring is a totally non-maximal order. These issues will be subject of further research. To avoid miss-interpretation of these result it should be noted that non-maximal orders, such as those applied in [19,26], where the factorization of Δ_p is kept secret, are not effected by this result.

References

- E. Bach, J. Shallit: Algorithmic number theory, vol. 1 Efficient Algorithms, Foundations of computing, MIT press, ISBN 0-262-02405-5, (1996)
- I. Biehl and J. Buchmann: "An analysis of the reduction algorithms for binary quadratic forms," Voronoi's Impact on Modern Science, vol. 1, Institute of Mathematics of National Academy of Sciences, Kyiv, Ukraine, (1998) 229
- 3. I. Biehl, S. Paulus and T. Takagi: "An efficient undeniable signature scheme based on non-maximal imaginary quadratic orders," Proceedings of Mathematics of Public Key Cryptography, Toronto, (1999) 223
- 4. R. Brent: ECM champs, ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Richard.Brent/champs.ecm 223
- J. Buchmann and S. Düllmann: "On the computation of discrete logarithms in class groups," Advances in Cryptology - CRYPTO '90, Springer, LNCS 773, (1991), pp.134-139 222
- J. Buchmann, S. Düllmann, and H.C. Williams: "On the complexity and efficiency of a new key exchange system," Advances in Cryptology - EUROCRYPT '89, Springer, LNCS 434, (1990), pp.597-616 222
- Z.I. Borevich and I.R. Shafarevich: Number Theory, Academic Press, New York, (1966) 224
- 8. J. Buchmann and H.C. Williams: "A key-exchange system based on imaginary quadratic fields," Journal of Cryptology, 1, (1988), pp.107-118 219, 220, 221
- J. Cowie, B. Dodson, M. Elkenbracht-Huizing, A.K. Lenstra, P.L. Montgomery and J. Zayer: "A worldwide number field sieve factoring record: on to 512 bits," Advances in Cryptology - ASIACRYPT'96, Springer, LNCS 1163, (1996), pp.382-394 222
- H. Cohen: A Course in Computational Algebraic Number Theory. Graduate Texts in Mathematics 138. Springer: Berlin, (1993) 224, 225
- 11. D. Coppersmith, A.M. Odlyzko and R. Schroeppel: "Discrete logarithms in GF(p)," Algorithmica, 1, (1986), pp.1-15 222
- 12. D.A. Cox: Primes of the form $x^2 + ny^2$, John Wiley & Sons, New York, (1989) 224, 225, 226, 227
- 13. W. Diffie and M. Hellman: "New directions in cryptography," IEEE Transactions on Information Theory 22, (1976), pp.472-492 222
- 14. S. Düllmann: Ein Algorithmus zur Bestimmung der Klassenzahl positiv definiter binärer quadratischer Formen, PHD-thesis, University of Saarbrücken, (1991)
- 15. D.M. Gordon: "Discrete logarithms in GF(p) using the number field sieve," SIAM Journal on Discrete Mathematics, **6**, (1993), pp.124-138 222
- J.L. Hafner and K.S. McCurley: "A rigorous subexponential algorithm for computation of class groups," Journal of the American Mathematical Society, 2, (1989), pp.837-850 219, 220, 222
- 17. L.K. Hua: Introduction to Number Theory. Springer-Verlag, New York, (1982)

- M. Hartmann, S. Paulus and T. Takagi: "NICE New Ideal Coset Encryption -," to appear in Workshop on Cryptographic Hardware and Embedded Systems.
- D. Hühnlein, M.J. Jacobson, S. Paulus and T. Takagi: "A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption," Advances in Cryptology - EUROCRYPT'98, LNCS 1403, Springer, (1998), pp.294-307 222, 223, 224, 225, 230
- D. Hühnlein, A. Meyer and T. Takagi: "Rabin and RSA analogues based on non-maximal imaginary quadratic orders," Proceedings of ICICS '98, ISBN 89-85305-14-X, (1998), pp.221-240 223
- D. Hühnlein: "Efficient implementation of cryptosystems based on non-maximal imaginary quadratic orders," T.R. TI-6, Technische Universtät Darmstadt, (1999), available at http://www.informatik.tu-darmstadt.de/TI/Veroeffentlichung/TR/ Welcome.html#1999 219, 220, 223, 226, 229
- 22. M.J. Jacobson Jr.: Subexponential Class Group Computation in Quadratic Orders, PhD thesis, Technische Universtät Darmstadt, to appear, (1999) 222
- 23. H.W. Lenstra: "On the computation of regulators and class numbers of quadratic fields," London Math. Soc. Lecture Notes, **56**, (1982), pp.123-150 **221**
- 24. A.K. Lenstra and H.W. Lenstra Jr. (eds.): The development of the number field sieve, Lecture Notes in Mathematics, Springer, (1993) 222
- National Institute of Standards and Technology (NIST): Digital Signature Standard (DSS). Federal Information Processing Standards Publication 186, FIPS-186, 19th May, (1994)
- S. Paulus and T. Takagi: "A new public-key cryptosystem over the quadratic order with quadratic decryption time," to appear in Journal of Cryptology. 220, 222, 228, 230
- 27. H.J.J. te Riele: Factorization of RSA-140 with the Number Field Sieve, posting in sci.crypt.research, February (1999) 222
- R. Rivest, A. Shamir and L. Adleman: "A method for obtaining digital signatures and public key-cryptosystems," Communications of the ACM,21, (1978), pp.120-126 222
- R.D. Silverman: "The multiple polynomial quadratic sieve," Math. Comp. 48, (1987), pp.329-229
- R.J. Schoof: "Quadratic Fields and Factorization," Computational Methods in Number Theory. Math. Centrum Tracts 155. Part II. Amsterdam, (1983), pp.235-286.
- D. Weber: "Computing discrete logarithms with quadratic number rings," Advances in Cryptology EUROCRYPT '98, LNCS 1403, Springer, 1998, pp. 171-183
 222

General Adversaries in Unconditional Multi-party Computation*

Matthias Fitzi, Martin Hirt, and Ueli Maurer

Department of Computer Science ETH Zurich CH-8092 Zurich, Switzerland, {fitzi,hirt,maurer}@inf.ethz.ch

Abstract. We consider a generalized adversary model for unconditionally secure multi-party computation. The adversary can actively corrupt (i.e. take full control over) a subset $D \subseteq P$ of the players, and, additionally, can passively corrupt (i.e. read the entire information of) another subset $E \subseteq P$ of the players. The adversary is characterized by a generalized adversary structure, i.e. a set of pairs (D, E), where he may select one arbitrary pair from the structure and corrupt the players accordingly. This generalizes the classical threshold results of Ben-Or, Goldwasser and Wigderson, Chaum, Crépeau, and Damgård, and Rabin and Ben-Or, and the non-threshold results of Hirt and Maurer.

The generalizations and improvements on the results of Hirt and Maurer are three-fold: First, we generalize their model by considering mixed (active and passive) non-threshold adversaries and characterize completely the adversary structures for which unconditionally secure multi-party computation is possible, for four different models: Perfect security with and without broadcast, and unconditional security (with negligible error probability) with and without broadcast. All bounds are tight. Second, some of their protocols have complexity super-polynomial in the size of the adversary structure; we reduce the complexity to polynomial. Third, we prove the existence of adversary structures for which no polynomial (in the number of players) protocols exist.

The following two implications illustrate the usefulness of these results: The most powerful adversary that is unconditionally tolerated by previous protocols among three players is the one that passively corrupts one arbitrary player; using our protocols one can unconditionally tolerate an adversary that either passively corrupts the first player, or *actively* corrupts the second or the third player.

Moreover, in a setting with arbitrarily many cheating players who want to compute an agreed function with the help of a trusted party, we can relax the trust requirement into this helping party: Without support from the cheating players the helping party obtains no information about the honest players' inputs and outputs.

Keyword: General adversaries, mixed model, multi-party computation, unconditional security.

^{*} Research supported by the Swiss National Science Foundation (SNF), project no. SPP 5003-045293.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 232–246, 1999. © Springer-Verlag Berlin Heidelberg 1999

1 Introduction

1.1 Secure Multi-Party Computation

Consider a set of n players who do not trust each other. Nevertheless they want to compute an agreed function of their inputs in a secure way. Security means achieving correctness of the result of the computation while keeping the players' inputs private, even if some of the players are corrupted by an adversary. This is the well-known secure multi-party computation problem, as first stated by Yao [Yao82].

As the first general solution to this problem, Goldreich, Micali, and Wigderson [GMW87] presented a protocol that allows n players to securely compute an arbitrary function even if an adversary actively corrupts any t < n/2 of the players and makes them misbehave maliciously. However, this protocol assumes that the adversary is computationally bounded. In a model with secure and authenticated channels between each pair of players (the secure-channels model), Ben-Or, Goldwasser, and Wigderson [BGW88], and Chaum, Crépeau, and Damgård [CCD88] proved that unconditional security is possible if at most t < n/2 of the players are passively corrupted, or alternatively, if at most t < n/3 of the players are actively corrupted. The bound t < n/3 for the active model was improved by Rabin and Ben-Or [RB89], Beaver [Bea91], and Cramer, Damgård, Dziembowski, Hirt, and Rabin [CDD+99] to t < n/2 by assuming the existence of a broadcast channel.

Secure multi-party computation can alternatively, and more generally, be seen as the problem of performing a task among a set of players. The task is specified by involving a trusted party, and the goal of the protocol is to replace the need for the trusted party. In other words, the functionality of the trusted party is shared among the players. Secure function evaluation described above can be seen as a special case of this more general setting. Most protocols described in the literature in the context of secure function evaluation also apply in the general context. This also holds for the protocols described in this paper.

1.2 General Adversaries

Ito, Saito, and Nishizeki [ISN87] and Benaloh and Leichter [BL88] introduced the notion of general (non-threshold) access structures for secret sharing. For a set P of players, an access structure Γ is the set of all subsets of P that are qualified to reconstruct the secret. Hirt and Maurer [HM97] transferred and adjusted this notion to the field of general multi-party computation: for a set P of players, an adversary structure $\mathcal Z$ is a set of all subsets of P that are tolerated to jointly cheat without violating the security of the computation. A multi-party computation protocol is called $\mathcal Z$ -secure if its security is not affected even if an adversary corrupts the players in one particular set in $\mathcal Z$.

1.3 Contributions

The main results of [HM97] state that in the passive model, every function can be computed unconditionally \mathcal{Z} -securely if and only if no two sets in \mathcal{Z} cover the full set P of players. In the active model, every function can be computed \mathcal{Z} -securely if and only if no three sets in \mathcal{Z} cover P. Assuming the existence of broadcast channels and allowing some negligible error probability, every function can be computed \mathcal{Z} -securely if and only if no two sets in \mathcal{Z} cover P.

We unify these models and introduce a new model in which the adversary may actively corrupt some players, and, at the same time, passively corrupt some additional players. The adversary is characterized by a generalized adversary structure, a set of classes (D, E) of subsets of the player set P (i.e. $D, E \subseteq P$), where the players of one specific class (D, E) in the adversary structure may be corrupted — actively for the players in D (disruption) and passively for the players in E (eavesdropping).

For example, the adversary structure $\mathcal{Z} = \{(\{p_1\}, \{p_2, p_3\}), (\{p_2\}, \{p_4\})\}$ describes an adversary that may *either* simultaneously corrupt player p_1 actively and the players p_2 and p_3 passively, or simultaneously corrupt player p_2 actively and player p_4 passively. Note that it is not known in advance which class of the structure will be corrupted by the adversary (and this is typically even not known at the end of the protocol).

For this unified model, the necessary and sufficient conditions for secure multi-party computation to be achievable for all functions are derived. In order to characterize these conditions, we introduce three predicates: Let P be a set of players and let \mathcal{Z} be an adversary structure for P. Then $Q^{(2,2)}(P,\mathcal{Z})$ is the predicate that is satisfied if and only if the players of no two classes in \mathcal{Z} cover the full set P of players, $Q^{(3,2)}(P,\mathcal{Z})$ is the predicate that is satisfied if and only if the players of no two classes in \mathcal{Z} together with the players in the active set of any other class in \mathcal{Z} cover P, and finally, $Q^{(3,0)}(P,\mathcal{Z})$ is the predicate that is satisfied if and only if the players in the active sets of any three classes in \mathcal{Z} do not cover P. Formally,

$$Q^{(2,2)}(P,\mathcal{Z}) \iff \forall (D_1, E_1), (D_2, E_2) \in \mathcal{Z} : D_1 \cup E_1 \cup D_2 \cup E_2 \neq P ,$$

$$Q^{(3,2)}(P,\mathcal{Z}) \iff \forall (D_1, E_1), (D_2, E_2), (D_3, E_3) \in \mathcal{Z} : D_1 \cup E_1 \cup D_2 \cup E_2 \cup D_3 \neq P ,$$

$$Q^{(3,0)}(P,\mathcal{Z}) \iff \forall (D_1, E_2), (D_2, E_2), (D_3, E_3) \in \mathcal{Z} : D_1 \cup D_2 \cup D_3 \neq P .$$

We characterize the necessary and sufficient conditions on the existence of unconditionally secure multi-party protocols according to three different cases:

- With or without broadcast channels, perfectly secure (without any probability of error) multi-party computation is achievable if and only if $Q^{(3,2)}(P, \mathbb{Z})$ is satisfied
- Given a broadcast channel, unconditionally secure (with negligible probability of error) multi-party computation is achievable if and only if $Q^{(2,2)}(P, \mathbb{Z})$ is satisfied.

– Without a broadcast channel, unconditionally secure multi-party computation is achievable if and only if both predicates $Q^{(2,2)}(P, \mathbb{Z})$ and $Q^{(3,0)}(P, \mathbb{Z})$ are satisfied.

Moreover, for all models we propose constructions that yield protocols with computation and communication complexity polynomial in the size of the adversary structure and linear in the size of the circuit computing the function, as opposed to the protocols of [HM97] that have super-polynomial complexity in those cases with error probability. Furthermore we show that this construction is optimal in the sense that there are adversary structures which *require* protocols with complexity at least polynomial in the size of the adversary structure (and hence potentially super-polynomial in the number of players).

1.4 Related Work

Active and passive corruptions within the same model was first considered by Galil, Haber, and Yung [GHY87] for threshold multi-party computation. Chaum [Cha89] proposed protocols that provide security with respect to an adversary that either passively or actively corrupts players up to given thresholds. Dolev, Dwork, Waarts, and Yung [DDWY93] proposed protocols and proved tight bounds for message transmission unconditionally secure in simultaneous presence of active and passive corruptions.

Fitzi, Hirt, and Maurer [FHM98] proposed multi-party protocols secure against mixed threshold adversaries. Based on the constructions of classical multi-party protocols [BGW88,RB89], they constructed new protocols for an adversary that simultaneously actively, passively, and fail-corrupts players up to given thresholds. However, as pointed out by Damgård [Dam99], their protocols for the perfect model (without error probability) do not achieve security for all thresholds within the claimed bounds. In contrast to their work that modified existing protocols in order to achieve the required properties, in this paper we use the technique of player simulation [HM97] with classical protocols [BGW88,RB89] as a basis.

Cramer, Damgård, and Maurer [CDM99] proved that for every adversary structure for which multi-party computation is feasible and for which there is an efficient linear secret-sharing scheme, efficient multi-party protocols exist. Smith and Stiglic [SS98] consider also uniquely active adversaries and propose protocols for the active model with broadcast. The efficiency of their protocols is polynomial in the size of a span program that computes the adversary structure, however in Section 4 we prove that for some adversary structures, every protocol requires complexity exponential in the number of players. This proof also applies to models with only passive or only active corruptions.

¹ Indeed, the tightness proofs for the perfect models in this paper contradict the results of [FHM98]. See [Dam99] for more details.

1.5 Outline

In Sect. 2 we formally define the models. The main results of the paper, the characterization of the exact conditions for secure multi-party protocols as well as the protocol constructions, are given in Sect. 3. In Sect. 4 we prove the existence of adversary structures for which no protocols with complexity polynomial in the number of players exist. Finally, some conclusions and open problems are mentioned in Sect. 5.

2 Definitions and Model

This section gives a formal definition of the model used in this paper.

2.1 Protocols

A processor can perform operations in a fixed finite field $(\mathcal{F},+,*)$, can select elements from this field at random, and can communicate with other processors over perfectly authenticated and confidential synchronous channels (secure channels model).²

A protocol π among a set P of processors is a sequence of statements. There are input and output statements, transmit statements, and computation statements. The latter include addition, multiplication, and random selection of field elements.

A multi-party computation specification (or simply called specification) formally describes the cooperation to be performed. Intuitively, a specification specifies the cooperation in an ideal environment involving a trusted party. Formally, a specification is a pair (π_0, τ) consisting of a protocol π_0 among a set P_0 of processors, and the name of a trusted processor $\tau \in P_0$.

A general approach to multi-party computation is to construct protocols for arbitrary specifications, or, more generally, to find a function (called *multi-party protocol generator*) that takes a specification as an input and outputs a protocol that securely computes the specification.

2.2 Adversaries and Adversary Structures

An adversary A is a program that actively corrupts a certain subset of the processors and, at the same time, passively corrupts another subset of the processors. To passively corrupt a processor means to be able to permanently read all variables of that processor. To actively corrupt a processor means to be able to take full control over the processor, in particular to read and write all its

² In contrast to the players mentioned in the introduction, a processor is considered to only perform the computation, where inputs and outputs are given from/to some other entity (e.g. a person). This distinction avoids misunderstandings when processors are simulated by multi-party protocols.

variables. The complexity of an adversary is not assumed to be polynomial and may be unlimited.

To restrict a class C=(D,E) to the set P' of processors, denoted $(D,E)|_{P'}$, means to intersect both sets of the class with P', i.e. $(D,E)|_{P'}=(D\cap P',E\cap P')$. To restrict a structure $\mathcal Z$ to the set P' of processors means to restrict all classes in the structure.

2.3 Security

For an adversary A, a protocol A-securely computes a specification if, whatever A does in the protocol, the same effect could be achieved by A (with a modified strategy, but with similar costs) in the specification [Can98,Bea91,MR98]. For an adversary structure \mathcal{Z} and a specification (π_0, τ) , a protocol π \mathcal{Z} -securely computes the specification (π_0, τ) if for every adversary A of class $C \in \mathcal{Z}$, the protocol π A-securely computes the specification (π_0, τ) . Whenever the specification is clear from the context, we also say that a protocol tolerates an adversary A (a structure \mathcal{Z}) instead of saying that the protocol A-securely (\mathcal{Z} -securely) computes the specification.

3 Complete Characterization of Tolerable Adversaries

The basic technique for constructing a protocol that tolerates a given adversary structure is to begin with a threshold protocol (e.g. one of the protocols of [BGW88,CCD88,RB89]) among a small number of processors and to simulate some of these processors by subprotocols among appropriate sets of other processors [HM97]. The idea behind this is that everything a processor has to perform during the protocol execution (such as communication with other processors and local computations) can be simulated by a multi-party computation

³ This definition implies that every adversary of a given class C' can also be considered as an adversary of every class C with $C' \subseteq C$.

protocol among a set of processors. If the adversary is tolerated by this simulation protocol then the simulated processor can be considered to be uncorrupted. This procedure of processor simulation can be applied recursively, i.e., the processors that participate in the simulation of a processor can again be simulated by an appropriate set of other processors, and so on.

The proofs given in this section are only sketched. Formal proofs based on simulator techniques can be given according to [Can98,Bea91,MR98]. Also, the proofs in this section are given with respect to a static adversary (i.e. an adversary that at the beginning of the protocol selects the processors to be corrupted), but they can be easily modified to apply to a model with an adaptive adversary (i.e. an adversary that consecutively corrupts processors during the computation, depending on the information gained so far, where the processors corrupted at any time must form an admissible class in the adversary structure).

3.1 Perfectly Secure Multi-Party Computation

The main result of this section, the tight bounds as well as the protocol construction, are stated in Theorem 1. This general result is based on a solution for all adversary structures \mathcal{Z} with $|\overline{\mathcal{Z}}| \leq 3$, which is given in the following lemma.

Lemma 1. A set P of processors can compute every function/specification perfectly \mathcal{Z} -securely if $Q^{(3,2)}(P,\mathcal{Z})$ and $|\overline{\mathcal{Z}}| \leq 3$. The computation and communication complexities are linear in the size of the specification.

Proof. Consider an arbitrary adversary structure \mathcal{Z} with $|\overline{\mathcal{Z}}| \leq 3$ that satisfies $Q^{(3,2)}(P,\mathcal{Z})$, and a specification (π_0,τ) . We show that for every such structure \mathcal{Z} there exists a subset of the processors that can compute the specification in a secure way. If $|\overline{\mathcal{Z}}| < 3$, then the condition $Q^{(3,2)}(P,\mathcal{Z})$ immediately implies that there is a processor $p \in P$ that is not contained in any class of $\overline{\mathcal{Z}}$ (i.e. $\mathcal{Z}|_{\{p\}} =$ $\{(\emptyset,\emptyset)\}$). Hence this processor cannot be corrupted by any admissible adversary, and one can simply replace the occurrence of the trusted party τ in the protocol π_0 of the specification by the name of this processor. Thus assume that $|\overline{\mathcal{Z}}|$ 3 and $\overline{Z} = \{(D_1, E_1), (D_2, E_2), (D_3, E_3)\}$. Condition $Q^{(3,2)}(P, Z)$ implies that there exists a processor $p_3 \in P$ with $p_3 \notin D_1 \cup E_1 \cup D_2 \cup E_2 \cup D_3$ (but potentially $p_3 \in E_3$). Hence this processor remains uncorrupted by an adversary of the first or the second class, and is (at most) passively corrupted by an adversary of the third class. By symmetry reasons, there exist processors p_1 and p_2 which can be corrupted at most passively and only by an adversary of the first or the second class, respectively. This means that every admissible adversary may corrupt none of the processors p_1 , p_2 , or p_3 actively and only at most one of them passively. Hence, these three processors can simulate the trusted party of the specification by using the protocol of [BGW88] (passive model) for three processors. The other processors (if any) are not involved in the simulation of the trusted party.

⁴ Although only a subset of the processors is involved in the multi-party computation, all the processors that have input must provide (i.e. secret-share) this input among the involved processors.

Theorem 1. A set P of processors can compute every function/specification perfectly Z-securely if $Q^{(3,2)}(P,Z)$ is satisfied. This bound is tight: if $Q^{(3,2)}(P,Z)$ is not satisfied, then there exist functions that cannot be computed perfectly Z-securely, even if a broadcast channel is available. The communication and computation complexities are polynomial in the size $|\overline{Z}|$ of the basis of the adversary structure and linear in the length of the specification.

Proof. Consider a set P of processors and a structure \mathcal{Z} for this set P such that $Q^{(3,2)}(P,\mathcal{Z})$ is satisfied, and an arbitrary specification (π_0,τ) . We recursively construct a \mathcal{Z} -secure protocol π :

The case $|\overline{Z}| \leq 3$ was treated in Lemma 1 (induction basis). Thus assume that $|\overline{Z}| \geq 4$, and that for all adversary structures with basis size strictly less than k there exists a secure protocol (induction hypothesis). We select some four-partition of \overline{Z} where the size of each set of the partition is at least $\lfloor |\overline{Z}|/4 \rfloor$. Let Z_1 , Z_2 , Z_3 , and Z_4 be the four unions of three distinct sets of the partition, each of them completed such that it is monotone. Since $|\overline{Z}| \geq 4$, the size $|\overline{Z}_i|$ of the basis of each such structure is strictly smaller than the size $|\overline{Z}|$ of the current structure basis, i.e. $|\overline{Z}_i| < |\overline{Z}|$ ($1 \leq i \leq 4$), and one can recursively construct protocols π_1 , π_2 , π_3 , and π_4 , each among the set P of processors, tolerating Z_1 , Z_2 , Z_3 , and Z_4 , respectively (hypothesis). The protocol π that tolerates Z can be constructed as follows:

First, one constructs a protocol among four "virtual" processors that computes the specification (π_0, τ) , tolerating an adversary that actively corrupts a single processor [BGW88] (active model). Then one simulates the four virtual processors by the recursively constructed protocols π_1, \ldots, π_4 , respectively. Since every adversary class is tolerated by at least three of the protocols π_1, π_2, π_3 , and π_4 (thus only one of the virtual processors in the main protocol is misbehaving), the resulting protocol tolerates all adversary classes in the adversary structure and, as claimed, the constructed protocol π is \mathbb{Z} -secure.

In order to analyze the efficiency of the protocols, we need the help of the following observation: The protocols of [BGW88] for the passive model with three processors and those for the active model with four processors have a constant "blow-up factor" b_p and b_a , respectively, i.e. for any specification of length l, the length of the protocol computing this specification is bounded by $b_p \cdot l$ in the passive model and by $b_a \cdot l$ in the active model.

In the construction given above, on each recursion level all involved processors are simulated by using protocols of [BGW88] (active case), except for the lowest level, where [BGW88] (passive case) is used. The simulations on each level can be performed independently, and every statement in the current protocol is affected by at most two such simulations (as at most two processors occur in one statement). Hence, the total blow-up of all simulations on a given level is bounded by b_a^2 (b_p^2 on the lowest level), and as the recursion depth of the construction is logarithmic in the number $|\overline{\mathcal{Z}}|$ of maximal sets in the adversary structure, the total blow-up is polynomial in $|\overline{\mathcal{Z}}|$.

 $^{^5}$ Indeed, almost every non-trivial function cannot be computed perfectly $\mathcal{Z}\text{-securely}.$

In order to prove the tightness of the theorem, assume an adversary structure \mathcal{Z} for which every function can be computed perfectly \mathcal{Z} -securely and suppose $Q^{(3,2)}(P,\mathcal{Z})$ is not satisfied. Then there exist three classes $(D_1,E_1),(D_2,E_2),(D_3,E_3)\in\mathcal{Z}$ with $D_1\cup E_1\cup D_2\cup E_2\cup D_3=P$, and (due to the monotonicity of \mathcal{Z}) with the sets D_1,E_1,D_2,E_2 and D_3 being pairwise disjoint.

One can construct a protocol for three processors \hat{p}_1 , \hat{p}_2 , and \hat{p}_3 , where \hat{p}_1 plays for all the processors in $D_1 \cup E_1$, \hat{p}_2 plays for those in $D_2 \cup E_2$, and \hat{p}_3 plays for those in D_3 . This new protocol is secure with respect to an adversary that passively corrupts either \hat{p}_1 or \hat{p}_2 , or actively corrupts \hat{p}_3 .

Assume that the specification requires to compute the logical AND of two bits x_1 and x_2 held by \hat{p}_1 and \hat{p}_2 , respectively, and assume for the sake of contradiction that a protocol for this specification is given. Let T denote the transcript of the broadcast channel of a run of that protocol (if no broadcast channel is available, let $T=\emptyset$), and let T_{ij} ($1 \le i < j \le 3$) denote the transcript of the channels between \hat{p}_i and \hat{p}_j . Due to the requirement of perfect privacy, \hat{p}_1 will not send any information about his bit x_1 over T_{12} or over T before he knows x_2 (if P_1 knows that $x_2=1$ he can reveal x_1). Similarly, \hat{p}_2 will not send any information about x_2 over T_{12} or over T before he knows x_1 . Hence the only escape from this deadlock would be to use \hat{p}_3 . However, as T_{12} and T jointly give no information about x_2 , a random misbehavior of an actively corrupted \hat{p}_3 (ignore all received messages and send random bits whenever a message must be sent) would with some (possibly negligible) probability make \hat{p}_1 receive the wrong output, contradicting the perfect security of the protocol.

3.2 Unconditionally Secure Multi-Party Computation

We prove the necessity of $Q^{(2,2)}$ for unconditionally secure multi-party computation in Lemma 2, and prove its sufficiency for the case that broadcast channels are available in Theorem 2. We then consider a model without broadcast and suggest a simple but surprising protocol among three processors for this model (Theorem 3). Finally, in Theorem 4, the tight bounds on the existence of unconditionally secure multi-party protocols in a model without broadcast are given. Note that all proposed protocols are efficient (polynomial in the number of maximal sets in the adversary structure), as opposed to the protocols for the unconditional model in [HM97].

Lemma 2. For every adversary structure \mathcal{Z} for a processor set P not satisfying $Q^{(2,2)}(P,\mathcal{Z})$, there exist functions/specifications that cannot be computed unconditionally \mathcal{Z} -securely. Even a broadcast channel does not help.

Proof. For the sake of contradiction, assume that for an adversary structure \mathcal{Z} for which $Q^{(2,2)}(P,\mathcal{Z})$ is not satisfied, there exists an unconditional \mathcal{Z} -secure protocol for every function. There exist two classes $(D_1, E_1), (D_2, E_2) \in \mathcal{Z}$ with $D_1 \cup E_1 \cup D_2 \cup E_2 = P$. Without loss of generality, assume that the four sets D_1 , E_1 , D_2 , and E_2 are pairwise disjoint. Then we can transform such a \mathcal{Z} -secure

protocol into a protocol among two processors \hat{p}_1 and \hat{p}_2 , where each processor plays for the processors in $D_1 \cup E_1$, and $D_2 \cup E_2$, respectively. The broadcast channel is not needed anymore (there are only two processors). This protocol is secure against passive corruption of one of the two processors, contradicting Theorem 2 of [BGW88].

Theorem 2. If a broadcast channel is available, a set P of processors can compute every function/specification unconditionally Z-securely if $Q^{(2,2)}(P,Z)$ is satisfied. This bound is tight: if $Q^{(2,2)}(P,Z)$ is not satisfied, then there exist functions that cannot be computed unconditionally Z-securely. The communication and computation complexities of the protocol are polynomial in the size $|\overline{Z}|$ of the basis of the adversary structure and linear in the length of the specification.

Proof. Consider a set P of processors and a structure \mathcal{Z} for this set P such that $Q^{(2,2)}(P,\mathcal{Z})$ is satisfied, and an arbitrary specification (π_0,τ) . We have to construct a \mathcal{Z} -secure protocol π for the set P of processors.

The case $|\overline{Z}| \leq 3$ is simple. Since we have $Q^{(2,2)}(P, \mathbb{Z})$, we have three processors p_1, p_2 , and p_3 , where p_i occurs in the *i*-th class of \mathbb{Z} , but does not occur in the other classes. The protocol of [RB89] for three processors requires a broadcast channel and provides unconditional security (with some negligible error probability) with respect to an adversary that actively corrupts a single processor (trivially, this processor may also be corrupted only passively). This protocol among the three processors p_1, p_2 , and p_3 is \mathbb{Z} -secure.

The case of a basis with at least four classes is treated along the lines of the construction in the proof of Theorem 1: First we select some four-partition of $\overline{\mathcal{Z}}$ and, by recursion, a protocol is constructed for each of the four unions of three subsets of the partition. Then, these four protocols are composed to a four-party protocol of [BGW88, active model].

The efficiency of this protocol can be analysed along the lines of the analysis given in the proof of Theorem 1. However, as the protocols of [RB89] that are used in the lowest level of the substitution tree provide some negligible error probability, special care is required in the analysis (cf. [HM97]). It follows immediately from the analysis in the proof of Theorem 1 that the protocol which results after applying all substitutions except for those on the lowest level, has polynomial complexity. But every statement of this protocol is expanded at most twice by all the remaining substitutions (once per involved processor), and each blow-up is polynomial, and hence the final protocol is also polynomial in the number $|\overline{Z}|$ of maximal sets in the adversary structure. This is in contrast to the protocols of [RB89] are used in each level of the simulation tree and hence their protocols have superpolynomial complexity.

The tightness of the theorem is given in Lemma 2.

Proposition 1. Let \mathcal{Z} be an adversary structure for the set P of processors, where one processor $p \in P$ does not occur in the active set of any class $C \in \mathcal{Z}$ (i.e. $\forall (D, E) \in \mathcal{Z} : p \notin D$). If there exists a \mathcal{Z} -secure protocol π in a model with

broadcast, then one can construct a Z-secure protocol π' for a model without broadcast. The complexity of π' is the same as the complexity of π .

Proof. Since there exists a processor $p \in P$ that cannot be actively corrupted by any admissible adversary, it is guaranteed that it follows the protocol. Hence, p can be used to simulate a broadcast channel. Instead of broadcasting a message, the message is sent to p which then sends this message to all processors in P. \square

Theorem 3. A set $P = \{p_1, p_2, p_3\}$ of three processors can compute every function/specification unconditionally securely with respect to an adversary that either passively corrupts p_1 or actively corrupts either p_2 or p_3 , i.e. \mathbb{Z} -securely for $\overline{\mathbb{Z}} = \{(\emptyset, \{p_1\}), (\{p_2\}, \emptyset), (\{p_3\}, \emptyset)\}.$

Proof. In order to compute an arbitrary specification, the protocol of [RB89] is applied. This protocol for three processors provides unconditional security (with negligible error probability) with respect to an adversary that may actively corrupt one arbitrary processor, but it assumes the existence of a broadcast channel. However, the processor p_1 does not occur in the active set of any class in \mathcal{Z} , so, by Proposition 1, we can transform the protocol with a broadcast channel to a protocol that does not assume a broadcast channel.

Theorem 4. A set P of processors can compute every function/specification unconditionally Z-securely if $Q^{(2,2)}(P,Z)$ and $Q^{(3,0)}(P,Z)$ are satisfied. This bound is tight: if $Q^{(2,2)}(P,Z)$ or $Q^{(3,0)}(P,Z)$ is not satisfied, then there exist functions that cannot be computed unconditionally Z-securely. The communication and computation complexities of the protocol are polynomial in the size $|\overline{Z}|$ of the basis of the adversary structure and linear in the length of the specification.

Proof. Consider a set P of processors and a structure \mathcal{Z} for this set P such that $Q^{(2,2)}(P,\mathcal{Z})$ and $Q^{(3,0)}(P,\mathcal{Z})$ are satisfied. The condition $Q^{(3,0)}$ implies the existence of an efficient secure protocol for broadcast [FM98], and hence the construction of the proof of Theorem 2 yields a \mathcal{Z} -secure protocol.

The necessity of $Q^{(2,2)}(P, \mathbb{Z})$ was proven in Lemma 2. Thus assume that $Q^{(2,2)}(P, \mathbb{Z})$ is satisfied but not $Q^{(3,0)}(P, \mathbb{Z})$, i.e. there exist three classes $(D_1, E_1), (D_2, E_2), (D_3, E_3) \in \mathbb{Z}$ with $D_1 \cup D_2 \cup D_3 = P$ (and D_1, D_2 , and D_3 pairwise disjoint). For the sake of contradiction, assume that for every function a \mathbb{Z} -secure multi-party protocol exists, hence in particular for the broadcast function. One can hence construct a broadcast protocol for the three processors \hat{p}_1 , \hat{p}_2 , and \hat{p}_3 (where each processor \hat{p}_1 , \hat{p}_2 , and \hat{p}_3 "plays" for the processors in one of the sets D_1 , D_2 , and D_3 , respectively), where the adversary is allowed to actively corrupt one of them, contradicting the result that broadcast for three processors is not possible if the adversary may actively corrupt one of the processors, even if a negligible error probability is tolerated [LSP82,KY].

Corollary 1. Using the help of a trusted party τ , a set P of n processors can compute every function/specification unconditionally securely with respect to an adversary that may actively corrupt any subset $S \subseteq P$ of size $|S| \le t$ (for a given

t > n/2). The trusted party τ obtains no information about the private inputs and outputs as long as less than n-t processors are actively corrupted.

Proof. We need to show that there are \mathcal{Z} -secure protocols for the set $P \cup \{\tau\}$ of processors, where $\mathcal{Z} = \{(D,E) \colon |D \cup E| \le t\} \cup \{(D,E \cup \{\tau\}) \colon |D \cup E| < n-t\}$. According to Theorem 4 it suffices to show that $Q^{(2,2)}(P \cup \{\tau\}, \mathcal{Z})$ and $Q^{(3,0)}(P \cup \{\tau\}, \mathcal{Z})$ are satisfied. Obviously, $Q^{(3,0)}(P \cup \{\tau\}, \mathcal{Z})$ holds since τ may not be actively corrupted.

In order to prove that $Q^{(2,2)}(P \cup \{\tau\}, \mathcal{Z})$ is satisfied, consider two arbitrary classes $(D_1, E_1), (D_2, E_2) \in \mathcal{Z}$. At least one of the classes must contain τ (else the classes cannot cover $P \cup \{\tau\}$), and this class has cardinality at most n-t. The other class has cardinality at most n-t (if it contains τ) or t (if it does not contain τ), and the condition t > n/2 implies that in either case the sum cardinality of both classes is at most n. There are n+1 processors in $P \cup \{\tau\}$, hence at least one processor does not occur in either class and $Q^{(2,2)}(P \cup \{\tau\}, \mathcal{Z})$ is satisfied.

4 Adversary Structures without Efficient Protocols

The goal of this section is, informally, to prove that there exists a family of adversary structures for which the length of every resilient protocol grows exponentially in the number of processors.

For a specification (π_0, τ) , a set P of processors, and an adversary structure \mathcal{Z} , let $\varphi((\pi_0, \tau), P, \mathcal{Z})$ denote the length of the shortest protocol π for P that \mathcal{Z} -securely computes (π_0, τ) . Furthermore, let (π_*, τ) denote the specification for the processors p_1 and p_2 that reads one input of both processors, computes the product and hands it to p_1 . Finally, let P_n denote the set $\{p_1, \ldots, p_n\}$ of processors.

The following theorem shows that there exists a family $\mathcal{Z}_2, \mathcal{Z}_3, \ldots$ of adversary structures for the sets P_2, P_3, \ldots of processors, respectively, such that $\varphi((\pi_*, \tau), P_n, \mathcal{Z}_n)$ grows exponentially in n.

Theorem 5. For all considered models there is a family $\mathbb{Z}_2, \mathbb{Z}_3, \ldots$ of admissible adversary structures for the sets P_2, P_3, \ldots of processors such that the length $\varphi((\pi_*, \tau), P_n, \mathbb{Z}_n)$ of the shortest \mathbb{Z}_n -secure protocol for (π_*, τ) grows exponentially in n.

In order to prove the theorem we need an additional definition: An admissible adversary structure \mathcal{Z} for the set P of processors is maximal if $Q^{(3,2)}(P,\mathcal{Z})$ is satisfied, but any adversary structure \mathcal{Z}' with $\mathcal{Z} \subseteq \mathcal{Z}'$ violates $Q^{(3,2)}(P,\mathcal{Z}')$.

Proof. The proof proceeds in three steps: First we prove that the number of maximal admissible adversary structures grows doubly-exponentially in the number n of processors. In the second step, we show that for the given specification

 (π_*, τ) , for every maximal admissible adversary structure a different protocol is required. Finally, we conclude that for some adversary structures the length of every secure protocol is exponential in the number of processors.

- 1. We exclusively consider adversary structures \mathcal{Z} that only contain classes with an empty active set, i.e. $\forall (D,E) \in \mathcal{Z} : D = \emptyset$. Hence, the necessary and sufficient conditions for the existence of multi-party protocols is that the passive sets of no two classes in \mathcal{Z} cover the full set P of processors. As a shorthand we write $E \in \mathcal{Z}$ instead of $(D,E) \in \mathcal{Z}$. Without loss of generality, assume that n = |P| is odd, and let m = (n+1)/2. Fix a processor $p \in P$, and consider the set B that contains all subsets of $P \setminus \{p\}$ with exactly m processors, i.e. $B = \{E \subseteq (P \setminus \{p\}) : |E| = m\}$. For each subset $B' \subseteq B$, we define $\mathcal{Z}_{B'}$ to be the adversary structure that contains all sets in B', plus all sets $E \subseteq P$ with |E| < m and $(P \setminus E) \notin B$. One can easily verify that $\mathcal{Z}_{B'}$ is admissible and maximal, and that for two different subsets $B', B'' \subseteq B$, the structures $\mathcal{Z}_{B'}$ and $\mathcal{Z}_{B''}$ are different. The size of B is $|B| = \binom{n-1}{m}$, hence there are $2^{\binom{n-1}{m}}$ different subsets B' of B, and thus doubly-exponentially many different maximal admissible adversary structures.
- 2. Let \mathcal{Z} be a maximal admissible adversary structure, and let π be a protocol that \mathcal{Z} -securely computes (π_*, τ) . For the sake of contradiction, assume that for some other maximal admissible adversary structure \mathcal{Z}' (where $\mathcal{Z}' \neq \mathcal{Z}$), the same protocol π \mathcal{Z}' -securely computes (π_*, τ) . Then π would $(\mathcal{Z} \cup \mathcal{Z}')$ -securely compute (π_*, τ) . However, since both \mathcal{Z} and \mathcal{Z}' are maximal admissible, $(\mathcal{Z} \cup \mathcal{Z}')$ is not admissible, and hence no such protocol exists. Hence, for each maximal admissible adversary structure \mathcal{Z} a different protocol π is required for securely computing (π_*, τ) .
- 3. There are doubly-exponentially many maximal admissible adversary structures, and for each of them, a different protocol is required, hence there are doubly-exponentially many different protocols. This concludes that some of these protocols have exponential length.

5 Conclusions and Open Problems

We have given a complete characterization of adversaries tolerable in unconditional multi-party computation in a generalized model where the adversary may actively corrupt some players and simultaneously passively corrupt some additional players. The characterization of the adversary is given by a set of pairs of subsets of the player set (rather than thresholds as in [Cha89,DDWY93,FHM98] or an adversary structure for either passive or active corruption [HM97,CDM99,SS98]). Moreover we have proposed constructions that, for any admissible adversary, yield secure protocols with communication and computation complexities polynomial in the size of the adversary structure. This improves on those protocols in [HM97] that have complexities super-polynomial in the size of the adversary structure.

For many scenarios, the protocols proposed in this paper tolerate strictly more powerful adversaries than are tolerated by any previous protocol. As a surprising example, the protocol for three players that unconditionally tolerates an adversary that passively corrupts one single player could be improved by tolerating that the adversary may corrupt one of two specific players even actively.

Finally, this paper has given a proof that there is a family of adversary structures which no protocol with complexities polynomial in the number of players exists for.

Besides active and passive player corruption, fail-corruption can be considered as a third fundamental type of player corruption, as treated in [GHY87,DDWY93,FHM98]. It is an open problem to characterize the tight conditions for unconditionally secure multi-party computation to be achievable with respect to a general adversary that may simultaneously perform active, passive and fail-corruptions. In the generalized adversary model this problem seems to be more involved than in the threshold model.

Acknowledgments

The authors would like to thank Ronald Cramer, Ivan Damgård, Yuval Ishai, and Stefan Wolf for many interesting discussions. Furthermore we would like to thank the anonymous referees for their useful comments on the paper.

References

- Bea91. D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, pp. 75–122, 1991. 233, 237, 238
- BGW88. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th ACM Symposium on the Theory of Computing (STOC)*, pp. 1–10, 1988. 233, 235, 237, 238, 239, 241
- BL88. J. C. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In *Advances in Cryptology CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*. Springer-Verlag, 1988. 233
- Can98. R. Canetti. Security and composition of multi-party cryptographic protocols. Manuscript, June 1998. Former (more general) version: Modular composition of multi-party cryptographic protocols, Nov. 1997. 237, 238
- CCD88. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In Proc. 20th ACM Symposium on the Theory of Computing (STOC), pp. 11–19, 1988. 233, 237
- CDD⁺99. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations with dishonest minority. In Advances in Cryptology — EUROCRYPT '99, Lecture Notes in Computer Science, 1999. 233
- CDM99. R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret sharing scheme. Manuscript, 1999. 235, 244

- Cha89. D. Chaum. The spymasters double-agent problem. In Advances in Cryptology CRYPTO '89, volume 435 of Lecture Notes in Computer Science, pp. 591–602. Springer-Verlag, 1989. 235, 244
- Dam99. I. Damgård. An error in the mixed adversary protocol by Fitzi, Hirt and Maurer. Available at http://philby.ucsd.edu/cryptolib.html, paper 99-03, 1999. 235
- DDWY93. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, Jan. 1993. 235, 244, 245
- FHM98. M. Fitzi, M. Hirt, and U. Maurer. Trading correctness for privacy in unconditional multi-party computation. In *Advances in Cryptology CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, 1998. 235, 244, 245
- FM98. M. Fitzi and U. Maurer. Efficient Byzantine agreement secure against general adversaries. In *Distributed Computing DISC '98*, volume 1499 of *Lecture Notes in Computer Science*, Sept. 1998. 242
- GHY87. Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In Advances in Cryptology CRYPTO '87, volume 293 of Lecture Notes in Computer Science, pp. 135–155. Springer-Verlag, 1987. 235, 245
- GMW87. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game
 — a completeness theorem for protocols with honest majority. In *Proc. 19th ACM Symposium on the Theory of Computing (STOC)*, pp. 218–229, 1987.
 233
- HM97. M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. In Proc. 16th ACM Symposium on Principles of Distributed Computing (PODC), pp. 25–34, Aug. 1997. 233, 234, 235, 237, 240, 241, 244
- ISN87. M. Ito, A. Saito, and T. Nishizeki. Secret sharing scheme realizing general access structure. In *Proceedings IEEE Globecom '87*, pp. 99–102. IEEE, 1987. 233
- KY. A. Karlin and A. C. Yao. Manuscript. 242
- LSP82. L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. ACM Transactions on Programming Languages and Systems, 4(3):382–401, July 1982. 242
- MR98. S. Micali and P. Rogaway. Secure computation: The information theoretic case. Manuscript, 1998. Former version: Secure computation, In *Advances in Cryptology CRYPTO '91*, volume 576 of *Lecture Note in Computer Science*, pp. 392–404, Springer-Verlag, 1991. 237, 238
- RB89. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. 21st ACM Symposium on the Theory of Computing (STOC)*, pp. 73–85, 1989. 233, 235, 237, 241, 242
- SS98. A. Smith and A. Stiglic. Multiparty computation unconditionally secure against Q^2 adversary structures. Manuscript, July 1998. 235, 244
- Yao82. A. C. Yao. Protocols for secure computations. In Proc. 23rd IEEE Symposium on the Foundations of Computer Science (FOCS), pp. 160–164. IEEE, 1982.
 233

Approximation Hardness and Secure Communication in Broadcast Channels*

Yvo Desmedt^{1,2} and Yongge Wang³

Department of Computer Science, Florida State University, Tallahassee Florida FL 32306-4530, USA

desmedt@cs.fsu.edu

² Department of Mathematics, Royal Holloway, University of London, UK
³ Center for Applied Cryptographic Research, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada ygwang@cacr.math.uwaterloo.ca

Abstract. Problems of secure communication and computation have been studied extensively in network models. Goldreich, Goldwasser, and Linial, Franklin and Yung, and Franklin and Wright have initiated the study of secure communication and secure computation in multi-recipient (broadcast) models. A "broadcast channel" (such as Ethernet) enables one processor to send the same message—simultaneously and privately to a fixed subset of processors. Franklin and Wright, and Wang and Desmedt have shown that if there are at most k malicious (Byzantine style) processors, then there is an efficient protocol for achieving probabilisticly reliable and perfectly private communication in a strongly nconnected network where $n \geq k+1$. While these results are unconditional, we will consider these problems in the scenario of conditional reliability, and then improve the bounds. In this paper, using the results for hardness of approximation and optimization problems, we will design communication protocols (with broadcast channels) which could defeat more faults than possible with the state of the art. Specifically, assuming certain approximation hardness result, we will construct strongly n-connected graphs which could defeat a k-active adversary (whose computation power is polynomially bounded) for k = cn, where c > 1 is any given constant. This result improves a great deal on the results of Franklin and Wright, and Wang and Desmedt.

1 Introduction

If two parties are connected by a private and authenticated channel, then secure communication between them is guaranteed. However, in most cases, many

^{*} Research partly supported by DARPA F30602-97-1-0205. However the views and conclusions contained in this paper are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advance Research Projects Agency (DARPA), the Air Force, of the US Government.

Most of the research was done when the authors were at the University of Wisconsin – Milwaukee.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 247–257, 1999. © Springer-Verlag Berlin Heidelberg 1999

parties are only indirectly connected, as elements of an incomplete network of private and authenticated channels. In other words they need to use intermediate or internal nodes. Achieving participants cooperation in the presence of faults is a major problem in distributed networks. The interplay of network connectivity and secure communication have been studied extensively (see, e.g., [2,4,5,10]). For example, Dolev [4] and Dolev, Dwork, Waarts, and Yung [5] showed that, in the case of k Byzantine faults, reliable communication is achievable only if the system's network is 2k+1 connected. Hadzilacos [10] has shown that even in the absence of malicious failures connectivity k+1 is required to achieve reliable communication in the presence of k faulty participants.

Goldreich, Goldwasser, and Linial [9], Franklin and Yung [7], and Franklin and Wright [6] have initiated the study of secure communication and secure computation in *multi-recipient (broadcast)* models. A "broadcast channel" (such as Ethernet) enables one participant to send the same message—simultaneously and privately—to a fixed subset of participants. Franklin and Yung [7] have given a necessary and sufficient condition for individuals to exchange private messages in broadcast models in the presence of passive adversaries (passive gossipers). For the case of active Byzantine adversaries, many results have been presented by Franklin and Wright [6]. Note that Goldreich, Goldwasser, and Linial [9] have also studied the fault-tolerant computation in the public broadcast model in the presence of active Byzantine adversaries.

There are many examples of broadcast channels. A simple example is a local area network like an Ethernet bus or a token ring. Another example is a shared cryptographic key. By publishing an encrypted message, a participant initiates a broadcast to the subset of participants that is able to decrypt it.

We will abstract away the concrete network structures and consider multicast graphs. Specifically, a multicast graph is just a graph G(V,E). A vertex $A \in V$ is called a neighbor of another vertex $B \in V$ if there is an edge $(A,B) \in E$. In a multicast graph, we assume that any message sent by a node A will be received identically by all its neighbors, whether or not A is faulty, and all parties outside of A's neighborhood learn nothing about the content of the message. The neighborhood networks have been studied by Franklin and Yung in [7]. They have also studied the more general notion of hypergraphs, which we do not need.

As Franklin and Wright [6] have pointed out, unlike the simple channel model, it is not possible to directly apply protocols over multicast lines to disjoint paths in a general multicast graph, since disjoint paths may have common neighbors. Franklin and Wright have shown that in certain cases the change from simple channel to broadcast channel hurts the adversary more than it helps, because the adversary suffers from the restriction that an incorrect transmission from a faulty processor will always be received identically by all of its neighbors.

It was shown [6] that if the sender and the receiver are strongly n-connected (that is, there are n paths with disjoint neighborhoods) and the malicious adversary can destroy at most k processors, then the condition n > k is necessary and sufficient for achieving efficient probabilisticly reliable and probabilisticly private communication. They also showed that there is an efficient protocol to achieve

probabilisticly reliable and perfectly private communication when $n > \lceil 3k/2 \rceil$. Recently, Wang and Desmedt [13] have shown that, indeed, the condition n > k is necessary and sufficient for achieving efficient probabilisticly reliable and perfectly private communication in broadcast channels.

Definition 1. Let A and B be two vertices on a multicast graph G(V, E). We say that A and B are strongly n-connected if there are n neighborhood disjoint paths p_1, \ldots, p_n between A and B, that is, for any $i \neq j \leq n$, p_i and p_j have no common neighbor (except A and B). In other words, for any vertex $v \in V \setminus \{A, B\}$, if there is a vertex u_1 on p_i such that $(v, u_1) \in E$, then there is no u_2 on p_j such that $(v, u_2) \in E$.

However, all these results are concerned with malicious adversaries with unlimited computational power. In this paper, we will consider the situation when the adversary's computational power is polynomial time bounded. Specifically, assuming certain approximation hardness result, we will construct strongly n-connected multicast graphs which could defeat a k-active adversary (whose computation power is polynomial time bounded) for k = cn, where c > 1 is any given constant. This result improves a great deal on the results of Franklin and Wright [6] (which are for unconditional reliability). To achieve this improvement we use some of the hardness results of Burmester, Desmedt, and Wang in [3].

The idea underlying our construction is that we will design strongly n-connected communication graphs in such a way that it is hard for the adversary to find the neighborhood disjoint n paths which is a witness to the strong n-connectivity. Hence the adversary does not know which processors to block (or control).

There have been many results (see, e.g., [1,12] for a survey) for hardness of approximating an **NP**-hard optimization problem within a factor c from "below". For example, it is hard to compute an independent set V' of a graph G(V, E) with the property that $|V'| \geq \frac{n}{c}$ for some given factor c, where n is the size of the maximum independent set of G. But for our problem, we are more concerned with approximating an **NP**-hard optimization problem from "above". For example, given a graph G(V, E), how hard is it to compute a vertex set V' of G with $|V'| \leq cn$ such that V' contains an optimal independent set of G, where n is the size of the optimal independent set of G? Burmester, Desmedt, and Wang have shown that this kind of approximation problem is also **NP**-hard. We will use this result to design strongly n-connected multicast graphs which is secure against an active adversary who can control cn vertices where c > 1 is some constant.

The organization of this paper is as follows. We first present in Section 2 some graph theoretic result which we will need in this paper. Section 3 surveys the model for commutation in broadcast channels. In Section 4 we demonstrate how to use strongly *n*-connected graphs with trapdoors to achieve reliable and private communication against active adversaries. In Section 5 we outline an

An independent set in a graph G(V, E) is a subset V' of V such that no two vertices in V' are joined by an edge in E.

approach to build strongly n-connected graphs with trapdoors. We conclude in Section 6 with remarks towards theoretical improvements and we present some open problems.

2 Optimization and Approximation

In this section we survey and introduce some graph theoretic results which will be used in later sections.

Definition 2. The independent set problem is:

Instance: A graph G(V, E) and a number n.

Question: Does there exist a vertex set $V_1 \subseteq V$ of size n such that any two nodes in V_1 are not connected by an edge in E?

Definition 3. Given a graph G(V, E), a vertex subset $V' \subseteq V$ is called neighborhood independent if for any $u, v \in V'$ there is no $w \in V$ such that both (u, w) and (v, w) are edges in E.

Definition 4. A vertex v in a graph G(V, E) is isolated if there is no edge adjacent to v, i.e., for all $w \in V$, $(v, w) \notin E$.

Theorem 1. Given a graph G(V, E) and a number n, it is **NP**-complete to decide whether there exists a neighborhood independent set $V_1 \subseteq V$ of size n.

Proof. It is clear that the specified problem is in **NP**. Whence it suffices to reduce the **NP**-complete problem IS (Independent Set) to our problem.

The input G(V, E), to IS, consists of a set of vertices $V = \{v_1, \ldots, v_m\}$ and a set of edges E. In the following we construct a graph $f(G) = GNI(V_G, E_G)$ such that there is an independent set of size n in G if and only if there is a neighborhood independent set of size n in GNI.

Let $V_G = V \cup V'$ where $V' = \{v_{i,j} : (v_i, v_j) \in E, i < j\} \cup \{v_{i,i} : v_i \text{ is an isolated vertex}\}$ and $E_G = \{(v_i, v_{i,j}), (v_{i,j}, v_j) : v_{i,j} \in V', i \le j\} \cup \{(v_{i,j}, v_{i',j'}) : v_{i,j}, v_{i',j'} \in V', i \le j, i' \le j'\}$. It is straightforward to check that, for any neighborhood independent set $V_1 \subseteq V_G$, if $V_1 \cap V' \ne \emptyset$ then $|V_1| = 1$. It is also clear that for any two vertex $u, v \in V$, u and v have no common neighbor in f(G) if and only if $(u, v) \notin E$. Hence there is a neighborhood independent set of size v in v

The following results follow directly from the corresponding results for independent sets in Burmester, Desmedt, and Wang [3].

Theorem 2. ([3]) There is a constant $\varepsilon > 0$ such that it is **NP**-hard to compute a vertex set $V' \subseteq V$ of a graph G(V, E), with the following properties:

- 1. $|V'| \leq nm^{\varepsilon}$, where n is the size of the maximum neighborhood independent set of G and m = |V|.
- 2. V' contains a neighborhood independent vertex set V'' such that $|V''| \geq \frac{n}{2}$.

Corollary 1. ([3]) There is a constant $\varepsilon > 0$ such that it is **NP**-hard to compute a vertex set $V' \subseteq V$ of a graph G(V, E), with the following properties:

- 1. $|V'| \leq nm^{\varepsilon}$, where n is the size of the maximum neighborhood independent set of G and m = |V|.
- 2. V' contains a neighborhood independent vertex set V'' such that |V''| = n.

3 Models

Following Franklin and Wright [6], we consider multicast as our only communication primitive. A message that is multicast by any node in a multicast neighbor network is received by all its neighbors with privacy (that is, non-neighbors learn nothing about what was sent) and authentication (that is, neighbors are guaranteed to receive the value that was multicast and to know which neighbor multicast it). In our models, we assume that all nodes in the multicast graph know the complete protocol specification and the complete structure of the multicast graph. In a message transmission protocol, the sender A starts with a message M^A drawn from a message space $\mathcal M$ with respect to a certain probability distribution. At the end of the protocol, the receiver B outputs a message M^B . We consider a synchronous system in which messages are sent via multicast in rounds. During each round of the protocol, each node receives any messages that were multicast by its neighbors at the end of the previous round, flips coins and perform local computations, and then possibly multicast a message.

Generally there are two kinds of adversaries. A passive adversary (or gossiper adversary) is an adversary who can only observe the traffics through k internal nodes. An active adversary (or Byzantine adversary) is an adversary with polynomial-time bounded computational power who can control k internal nodes. That is, an active adversary will not only listen to the traffics through the controlled nodes, but also control the message sent by those controlled nodes. Both kinds of adversaries are assumed to know the complete protocol specification, message space, and the complete structure of the multicast graph. At the start of the protocol, the adversary chooses the k faulty nodes. A passive adversary can view the behavior (coin flips, computations, message received) of all the faulty nodes. An active adversary can view all the behavior of the faulty nodes and, in addition, control the message that they multicast. We allow for the strongest adversary. Throughout this paper, unless specified otherwise, we will use k to denote the number of nodes that the adversary can control and use k to denote the connectivity of the network.

For any execution of the protocol, let adv be the adversary's view of the entire protocol. We write adv(M,r) to denote the adversary's view when $M^A=M$ and when the sequence of coin flips used by the adversary is r.

Definition 5. (see Franklin and Wright [6])

1. A message transmission protocol is δ -reliable if, with probability at least $1-\delta$, B terminates with $M^B=M^A$. The probability is over the choices of M^A and the coin flips of all nodes.

- 2. A message transmission protocol is ε -private if, for every two messages M_0, M_1 and every $r, \sum_c |\Pr[adv(M_0, r) = c] \Pr[adv(M_1, r) = c]| \leq 2\varepsilon$. The probabilities are taken over the coin flips of the honest parties, and the sum is over all possible values of the adversary's view.
- 3. A message transmission protocol is perfectly private if it is 0-private.
- 4. A message transmission protocol is (ε, δ) -secure if it is ε -private and δ reliable.
- 5. An (ε, δ) -secure message transmission protocol is efficient if its round complexity and bit complexity are polynomial in the size of the network, $\log \frac{1}{\varepsilon}$ (if $\varepsilon > 0$) and $\log \frac{1}{\lambda}$ (if $\delta > 0$).

In order for an adversary to attack the broadcast communication system which is modeled by a strongly n-connected graph, s/he does not need to find all of the n neighborhood disjoint paths $\{p_1, \ldots, p_n\}$ in the graph. S/he can choose to control one neighbor vertex on each of the n paths. We therefore give the following definition.

Definition 6. Let G be a strongly n-connected graph, and $P = \{p_1, \ldots, p_n\}$ be a witness to the strong n-connectivity of the graph. A set $S = \{v_1, \ldots, v_k\}$ of vertices in G is called an eavesdropping vertex set of P if for each path $p_i \in P$ $(i = 1, \ldots, n)$, there is a $v_j \in S$ which is a neighbor of (at least one of the vertices in) p_i .

Note that in the above definition, k could be larger than or smaller than n.

Remark 1. The problem of finding an eavesdropping vertex set in a strongly k-connected graph is **NP**-hard which will be proved in Section 5.

The following theorem follows straightforwardly from the corresponding theorems in Franklin and Wright [6], and Wang and Desmedt [13].

Theorem 3. (Franklin and Wright [6], Wang and Desmedt [13]) If A and B are strongly n-connected, and the adversary does not control an eavesdropping vertex between A and B, then there is an efficient $(0, \delta)$ -secure message transmission protocol between A and B.

4 Reliable and Private Communication with Trapdoors

In this section, we show how to design reliable and private communication systems with trapdoors such that the following condition is satisfied:

– The broadcast communication system modeled by a strongly n-connected graph is robust against a polynomial time bounded k-active adversary where $k \leq cn$ and c > 1 is any given constant.

The idea is to use the fact that it is NP-hard to find an eavesdropping vertex set of a strongly n-connected graph. It follows that if one designs the graph in such a way that the trusted participants can easily find a witness to the

strong n-connectivity of the graph, and the sender and receiver always initiate a communication through this witness, then reliable and private communication is possible. The benefit from using trapdoors in a communication system with broadcast channels is obvious. If we do not use trapdoors then, Franklin and Wright [6]'s results show that a strongly n-connected graph is only robust against k-active adversaries when k < n. However, if we use trapdoors in the design of graphs, then with high probability, a strongly n-connected graph is robust against k-active adversaries where $k \le cn$ and c > 1 is any given constant. The reason is that even though the adversary has the power to jam or control k > n vertices in the graph, he does not know which vertices to corrupt such that each path p_i will have a corrupted neighbor (which can eavesdrop on the messages sent through the path p_i).

Definition 7. Let $\{\mathcal{G}_n\}_{n\in\mathcal{N}}$ be an ensemble of graphs with the property that each graph in \mathcal{G}_n is strongly n-connected but not strongly n+1-connected, where \mathcal{N} is the set of positive integers, and let k_n $(n=1,2,\ldots)$ be a sequence of positive integers. The ensemble $\{\mathcal{G}_n\}_{n\in\mathcal{N}}$ is called polynomial-time robust against k_n -active adversaries if for every probabilistic polynomial-time algorithm D with the property that for each $G \in \mathcal{G}_n$, D(G) is a k_n -element vertex subset of G, and for every polynomial $p(\cdot)$ and all sufficiently large n, the following condition is satisfied:

If G_n is not empty then the following inequality holds:

$$\left| \sum_{G \in \mathcal{G}_n} \operatorname{Prob} \left(D(G) \text{ is an eavesdropping vertex set of } G \right) \right| < \frac{1}{p(n)}.$$

The probabilities in the above definition are taken over the corresponding random variables \mathcal{G}_n and the internal coin tosses of the algorithm D.

Indeed, for the k_n and n in Definition 7, if $k_n < n$, then every ensemble $\{\mathcal{G}_n\}_{n \in \mathcal{N}}$ is polynomial-time robust against k_n -active adversaries. So one of the main problems is to design graph ensembles $\{\mathcal{G}_n\}_{n \in \mathcal{N}}$ which are polynomial-time robust against k_n -active adversaries for $k_n \geq n$, that is, to design strongly n-connected graphs in which it is hard on the average case to find an eavesdropping vertex set of size $k_n \geq n$. In Section 5, we will outline an approach to generate such kind of graphs. In the remaining part of this section we will demonstrate how to use these graphs to achieve reliability and privacy in broadcast channels.

Protocol I

- 1. Alice generates a strongly n-connected graph G such that finding a size k(=cn) eavesdropping vertex set is hard, where c>1 is any given constant. (The details will be presented in Section 5).
- 2. Using a secure channel, Alice sends to the sender and the receiver the n neighborhood disjoint paths $P = \{p_1, \ldots, p_n\}$ which is a witness to the strong n-connectivity of G.

3. In order to carry out one communication, the sender and the receiver initiate the communication protocol in Theorem 3 through the n paths in P.

Note that our above protocol is not proactive, that is, it is not secure against a dynamic adversary who after observing one communication will change the vertices he controls. Indeed, it is an interesting open problem to design protocols which are secure against dynamic adversaries.

First assume that Mallory is a k-active adversary where k = cn for some constant c > 1, and $P = \{p_1, \ldots, p_n\}$ is the set of neighborhood disjoint paths used in Protocol I. Since Mallory does not know how to find a size k eavesdropping vertex set for P (finding such a set is very hard, e.g., as hard as factoring, let say a 1024-bit integer), she does not know which vertices to corrupt so that she can corrupt the system even though she has the power to corrupt k = cn vertices. It follows that the system is robust against a k-active adversary.

5 Strongly *n*-connected Graphs with Trapdoors

In this section, we consider the problem of designing strongly n-connected graphs with trapdoors. By using Corollary 1, we will construct practical, average-case hard, strongly n-connected, graphs which are robust against k-active adversaries for k=n+c, where c is some given constant. Our following construction is based on the hardness of factoring a large integer and we will not use the approximation hardness results (which will be used to prove theoretical results in the next section).

Construction Let N be a large number which is a product of two primes p and q. We will construct a strongly n-connected graph G with the following property: given the number N and an eavesdropping vertex set for G, one can compute efficiently the two factors p and q. Let x_1, \ldots, x_t and y_1, \ldots, y_t be variables which take values 0 and 1, where $t = \lfloor \log N \rfloor$. And let $(x_t \ldots x_1)_2$ and $(y_t \ldots y_1)_2$ denote the binary representations of $\sum x_i 2^{i-1}$ and $\sum y_i 2^{i-1}$ respectively. Then use the relation

$$(x_t \dots x_1)_2 \times (y_t \dots y_1)_2 = N \tag{1}$$

to construct a 3SAT formula C with the following properties (the details of the construction are omitted. Indeed, one can use the constructive proof that 3SAT is **NP**-complete (see, e.g., [8, pp. 48-49]) to construct the 3SAT formula C though there are more efficient ways for our construction):

- 1. C has at most $O(t^2)$ clauses.
- 2. C is satisfiable and, from a satisfying assignment of C, one can compute in linear time an assignment of $x_1, \ldots, x_t, y_1, \ldots, y_t$ such that the equation (1) is satisfied. That is, from a satisfying assignment of C, one can factor N easily.

Now, by combining the construction in [8, pp. 48-49] (which constructs a graph GI for each 3SAT formula C with the property that GI has an independent set of size l for some constant $l = O(t^2)$ if and only if C is satisfiable) and the reduction in the proof of Theorem 1, construct a graph G'(V', E') and a number $n = O(t^2)$ with the property that: from a size n neighborhood independent set of G', one can compute in linear time a satisfying assignment of C. Lastly, the following procedure will generate a strongly n-connected graph G with the property that, from a size n + c eavesdropping vertex set of G, one can compute in linear time a size n neighborhood independent set of G'. Whence from any size n + c eavesdropping vertex set of G, one can compute in polynomial time the primes p and q. As a summary, our construction proceeds as follows.

$$(N, p, q) \to \text{graph } G' \to \text{strongly } n\text{-connected graph } G$$

Procedure for generating G from G'(V', E'): In the following we construct a multicast graph f(G') = G(V, E) and two nodes $A, B \in V$ (where A denotes the sender and B denotes the receiver) such that there is a neighborhood independent set of size n in G' if and only if A and B are strongly n-connected.

Let $V = \{A, B\} \cup V'$, and $E = E' \cup \{(A, v), (v, B) : v \in V'\}$. It is clear that two paths $P_1 = (A, v_i, B)$ and $P_2 = (A, v_j, B)$ are vertex disjoint and have no common neighbor (except A and B) in G if and only if v_i and v_j have no common neighbor in G'(V', E'). Hence there is a neighborhood independent set of size n in G' if and only if A and B are strongly n-connected in G. It is now sufficient to show that from each size n + c eavesdropping vertex set S' of G, one can compute in polynomial time a size n neighborhood independent set of G'.

Since S' is an eavesdropping vertex set of G and G is strongly n-connected, there is at least one size n subset S of S' such that

- -S itself is an eavesdropping vertex set of G;
- S is a neighborhood independent set of G'.

There are $\binom{n+c}{n} = \binom{n+c}{c}$ (which is a polynomial in n) many different size n subsets of S'. Whence by considering all these different size n subset of S' we can compute in polynomial time a size n vertex set S with the above properties.

It is straightforward to see that the above constructed strongly n-connected graph G is robust against k-active adversaries for k=n+c if factoring N is hard, where c is any given constant.

In order to state our main theorem, we need the following assumption of average hardness of factoring.

Hardness Assumption of Factoring: There exists an ensemble $\{X_n\}_{n\in\mathcal{N}}$ (where X_n is a subset of composite numbers of length n) such that for every probabilistic polynomial-time algorithm D from positive integers to positive integers, every polynomial $p(\cdot)$, and all sufficiently large n, the following condition is satisfied:

$$\left| \sum_{x \in X_n} Prob\left(D(x) \text{ is a non-trivial factor of } x\right) \right| < \frac{1}{p(n)}.$$

Now it is clear that our above discussion implies the following result:

Theorem 4. Assume the average hardness of factoring, then we can construct a graph ensemble $\{\mathcal{G}_n\}_{n\in\mathcal{N}}$ which is polynomial-time robust against k_n -active adversaries, where $k_n = n + c$ for some constant c > 1.

Proof. It follows from the preceding discussions.

6 Towards Theoretical Improvements

In the previous section, we outlined a "practical" approach for constructing strongly n-connected graphs which are robust against k-active adversaries for k = n + c. In this section we consider theoretical improvements. That is, we will construct strongly n-connected graphs which are robust against k-active adversaries for k = cn.

Construction First generate a graph G'(V', E') and a number n which satisfy the conditions of Corollary 1. Secondly, using the method from the previous section of constructing the strongly n-connected graph G(V, E) from G'(V', E'), we construct the strongly n-connected graph G(V, E) with the following properties:

- 1. Two paths P_1 and P_2 in G which go through u_i and u_j respectively are neighborhood disjoint if and only if u_i and u_j have no common neighbor in G' (see the previous section for details).
- 2. There is a size n neighborhood independent set in G' if and only if there are n neighborhood disjoint paths in G. And from n neighborhood disjoint paths in G one can compute in linear time a size n neighborhood independent set in G'.

From the construction of G from G', it is straightforward that for any size cn eavesdropping vertex set S' of G, S' contains a size n neighborhood independent set of G'.

By Corollary 1, the graph G is robust against cn-active adversaries.

Remark 2. The above construction shows that, with some reasonable assumption (for example, assume the existence of a probabilistic polynomial time algorithm to generate hard strongly n-connected graphs needed in the above construction, it is possible to construct an ensemble $\{\mathcal{G}_n\}_{n\in\mathcal{N}}$ of strongly n-connected graphs which is robust against polynomial-time k_n -active adversaries, where $k_n=cn$ for some constant c>1.

In this section, we constructed strongly n-connected graphs which are robust against cn-active adversaries. However, these constructions are inefficient and are only of theoretical interests, since the size of the graph G in Corollary 1 will be enormous if we want to make the security of the system to be at least as hard as an exhaustive search of a 1024-bit space. One of the most interesting open

questions is how to efficiently generate hard instances of strongly n-connected graphs, especially, for arbitrary number n.

We should also note that, in order to construct the strongly n-connected graphs in this section, we need to construct standard graphs which satisfy the conditions of Corollary 1. That is, we need an algorithm to build graphs whose neighborhood independent sets are hard to approximate in the average case. Whence it is interesting (and open) to prove some average-case hardness results for the corresponding problems.

Our protocols in this paper are not proactive, that is, not robust against a dynamic adversary who after observing one communication will change the vertices he controls. It is an interesting open problem to design protocols which are secure against dynamic adversaries.

References

- S. Arora. Probabilistic Checking of Proofs and Hardness of Approximation Problems. PhD Thesis, CS Division, UC Berkeley, August, 1994. 249
- M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computing. In: Proc. ACM STOC '88, pages 1–10, ACM Press, 1988. 248
- M. Burmester, Y. Desmedt, and Y. Wang. Using approximation hardness to achieve dependable computation. In: Proc. of the Second International Conference on Randomization and Approximation Techniques in Computer Science, LNCS 1518, pages 172–186, Springer Verlag, 1998. 249, 250, 251
- 4. D. Dolev. The Byzantine generals strike again. J. of Algorithms, ${\bf 3},$ pp. 14–30, 1982. 248
- D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. J. of the ACM, 40(1), pp. 17–47, 1993.
- M. Franklin and N. Wright. Secure communication in minimal connectivity models. In: Advances in Cryptology, Proc. of Eurocrypt '98, LNCS 1403, pages 346–360, Springer Verlag, 1998. 248, 249, 251, 252, 253
- M. Franklin and M. Yung. Secure hypergraphs: privacy from partial broadcast. In: Proc. ACM STOC '95, pages 36–44, ACM Press, 1995.
- M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, San Francisco, 1979. 254, 255
- O. Goldreich, S. Goldwasser, and N. Linial. Fault-tolerant computation in the full information model. SIAM J. Comput. 27(2):506–544, 1998.
- V. Hadzilacos. Issues of Fault Tolerance in Concurrent Computations. PhD thesis, Harvard University, Cambridge, MA, 1984.
- 11. J. Lewis. On the complexity of the maximum subgraph problem. In: *Proc. ACM STOC '78*, pages 265–274, ACM Press, 1978.
- M. Sudan. Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems. PhD. thesis, U. C. Berkeley, 1992.
- Y. Wang and Y. Desmedt. Secure communication in broadcast channels: the answer to Franklin and Wright's question. In: Advances in Cryptology, Proc. of Eurocrypt '99, LNCS 1592, pages 443–455, Springer Verlag, 1999. 249, 252

Mix-Networks on Permutation Networks

Masayuki ABE

NTT Laboratories
Nippon Telegraph and Telephone Corporation
1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan
abe@isl.ntt.co.jp

Abstract. Two universally verifiable mix-net schemes that eliminate the cumbersome Cut-and-Choose method are presented. The construction is based on a permutation network composed of a network of 'switches' that transposes two inputs. For N inputs and t tolerable corrupt mix-servers, the schemes achieve $\mathcal{O}(tN\log N)$ efficiency in computation and communication while previous schemes require $\mathcal{O}(\kappa mN)$ for error probability $2^{-\kappa}$ and m mix-servers. The schemes suit small to middle-scale secret-ballot electronic elections. Moreover, one of the schemes enjoys less round complexity so that servers do not need to talk to other servers except their neighbors unless disruption occurs.

1 Introduction

Mix-net is a cryptographic primitive that consists of a series of mix-servers that decrypt input ciphertexts and output the results in random order; thus it cuts the link between inputs and outputs. It has been mainly used to construct electronic secret ballot voting schemes where privacy should be protected.

Early constructions e.g., [5] and [13] are weak against the active deviation of mix-servers. For instance, any one server can control the output by stealthily replacing messages unless every user makes sure that his/her message appears among the output. In [15], Sako and Killian introduced a universally verifiable scheme by which the correctness of the entire output could be verified by anybody [15]. Robustness was first added in [12]. These later schemes are based on the cumbersome Cut-and-Choose method where each server and verifier suffers $\mathcal{O}(\kappa mN)$ computation and communication cost for m servers, N inputs, and error probability $1/2^{\kappa}$. In [1], the cost for a verifier was reduced to $\mathcal{O}(\kappa N)$ at the cost of an increase in communication between servers.

In [10,11], Jakobsson introduced robust schemes that are efficient in computation for large N. However, unlike the previous universally verifiable schemes, a verifier need to trust at least one server to convince himself of the correctness. Such a model would be sufficient when managers of mix-servers are selected from parties which can discredit each other. Thus, for instance, such schemes suit large-scale election where no conspiracy is expected. On the contrary, this paper addresses universally verifiable schemes where the correctness can be verified without trusting any server. Such a property prevents managers of mix-servers

from being blackmailed or committing bribery because the final output is out of their control.

Another drawback of previously known robust schemes is that the servers must interact each other frequently to verify intermediate results. If such schemes are implemented on a real network such as the Internet where the cost is excessive in terms of overhead, round complexity may dominate the total efficiency.

Our contribution: This paper presents two efficient robust and universally verifiable mix-nets, referred as MiP-1 and MiP-2, that are the first to eliminate Cut-and-Choose. The heart of our construction is the use of *Permutation Networks* where inputs can be arbitrarily permuted according to a control signal. Both of our schemes offer $\mathcal{O}(t\,N\,\log N)$ computation/communication efficiency where t is the number of tolerable corrupt servers. As complexity grows faster in N than the previous schemes, our solution suits small to moderate scale applications (but covers many practical cases). For instance, for $N < 2^{25}$, both schemes require less computation than the scheme in [12] at $\kappa = 80$ and m = 5. Furthermore, MiP-1 has an attractive property in that servers only need to send data to the next server unless disruption happens.

2 Model

Participants: Users $\{U_i\}_{i\in\{1,...,N\}}$, mix-servers $\{M_j\}_{j\in\{1,...,m\}}$, and a verifier V. All of them are polynomially bounded.

Communication Channel: We assume the use of a bulletin board BB where participants read and write in authenticated manner. No one can cancel any information once written to the BB.

Adversary: It is assumed that there exists a polynomially bounded adversary A who can corrupt up to some fixed number of mix-servers and users. We denote the set of corrupt users and servers by \mathcal{U}_A and \mathcal{M}_A respectively. Similarly, \mathcal{U}_H and \mathcal{M}_H denote honest users and servers respectively. A controls \mathcal{U}_A and \mathcal{M}_A in an arbitrary way to break anonymity or to create incorrect output.

Security: Here we present a sketch of the definitions of security properties. Let $E_y(msg,r)$ denote the probabilistic encryption of message msg with random factor r and encryption key y. Let $D_x()$ be the corresponding decryption function. The space for plain messages defined by the encryption function is denoted by MSG_E . An application may define its own message space $\Sigma \subseteq MSG_E$. Let (E_1, \ldots, E_N) denote an input to a mix-net, and (v_1, \ldots, v_N) denote the output obtained as a result of invocation of Mix-net protocols with respect to the input.

Definition 1 (Correctness). The output (v_1, \ldots, v_N) is correct if there exists a permutation between (v_1, \ldots, v_N) and $(D_x(E_1), \ldots, D_x(E_N))$.

Definition 2 (Universal Verifiability). Let $view_{pub}$ be all the information written in the BB during an execution of the mix-net protocols. A mix-net is verifiable if there exists a (probabilistic) polynomial-time algorithm V that, on

input view_{pub}, outputs accept if (v_1, \ldots, v_N) is correct, otherwise outputs reject with overwhelming probability.

Let $\mu_i \in \Sigma$ be the plaintext chosen by user U_i . Σ_H denotes the choice of honest users, i.e., $\Sigma_H = \{\mu_i \mid \mathsf{U}_i \in \mathcal{U}_H\}$. Let \mathcal{M}_I be an ideal-model mix (oracle) that takes (μ_1, \ldots, μ_N) as inputs from the users via authenticated untappable channels and outputs (v_1, \ldots, v_N) that satisfies $v_i = \mu_{\pi(i)}$ for all i and some permutation π on $\{1, \ldots, N\}$. Let AI be an ideal-model adversary such that given \mathcal{M}_I and \mathcal{U}_H as black boxes, outputs (i, μ) that satisfies $\mathsf{U}_i \in \mathcal{U}_H$ and $\mu \in \Sigma_H$. Similarly, let A be a real-life adversary such that given \mathcal{M}_H and \mathcal{U}_H as black boxes, outputs (i, μ) that satisfies $\mathsf{U}_i \in \mathcal{U}_H$ and $\mu \in \Sigma_H$.

Definition 3 (Anonymity). A mix-net is (t_u, t_m) -anonymous if, for all polynomially bound ideal-model adversaries AI that can corrupt at most t_u users and for all polynomially bound real-life adversaries A that can corrupt at most t_u users and t_m mix-servers,

$$\left| Pr[(i,\mu) \leftarrow \mathcal{A}^{\mathcal{M}_H,\mathcal{U}_H} : \mu = D_x(E_i)] - Pr[(i,\mu) \leftarrow \mathcal{A}\mathcal{I}^{\mathcal{M}_I,\mathcal{U}_H} : \mu = \mu_i] \right|$$

is negligible.

Remark: The above definition does not cover the case where all users are in \mathcal{U}_A because, then, there is no anonymity to maintain.

Definition 4 (Robustness). A mix-net is (t_u, t_m) -robust if, for any adversary A that controls \mathcal{U}_A and \mathcal{M}_A that satisfy $|\mathcal{U}_A| \leq t_u$ and $|\mathcal{M}_A| \leq t_m$, \mathcal{M}_H stops in polynomial-time with correct output with overwhelming probability.

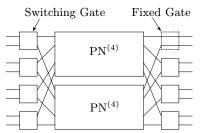
3 Overview

The clear structural difference between our two proposals is whether they use two phases (permutation phase, decryption phase) or one phase (randomized decryption phase). The two-phase scheme, referred to as MiP-2, bears some similarity to the schemes in [12,1] in that at the end of the first phase all servers must agree on whether they proceed to the second phase or not. This contrasts with the one-phase scheme, referred to as MiP-1, where no such synchronization is needed as long as no disruption occurs. Let us begin with MiP-2 which is rather easy to comprehend.

3.1 Sketch of MiP-2

MiP-2 consists of two phases: (1) the permutation phase in which inputs are randomized and randomly permuted, and (2) the decryption phase in which the result of the first phase is decrypted.

Let us first review the permutation network which is the heart of the permutation phase. A permutation network is a circuit which, on input (1, ..., N) and an



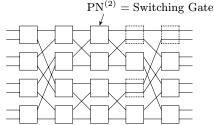


Fig. 1. $PN^{(8)}$ with $PN^{(4)}$

Fig. 2. $PN^{(8)}$ after decomposing $PN^{(4)}$

arbitrary permutation $\Pi: \{1, \ldots, N\} \to \{1, \ldots, N\}$, outputs $(\Pi(1), \ldots, \Pi(N))$. Let $PN^{(N)}$ denote a permutation network with N inputs hereafter. In this paper, we consider N to be a power of 2. A *switching gate* is a permutation network for two inputs, that is $PN^{(2)}$. It has two pairs of input/output terminals labeled by I_0 , I_1 , O_0 and O_1 . It also accepts control signal $b \in \{0,1\}$. According to the control signal, it outputs either $(O_0, O_1) = (I_0, I_1)$ or $(O_0, O_1) = (I_1, I_0)$. Suppose a switching gate provides delay of 1. The following theorem holds.

Theorem 5. For any N of power of 2, there exists $PN^{(N)}$ that consists of $N \log_2 N - N + 1$ switching gates and provides $2 \log_2 N - 1$ delay.

Please refer [17] for proof and detailed construction. Here we use intuitive figures to illustrate the recursive construction of $PN^{(8)}$. Dotted boxes indicate fixed gates that simply output the inputs. We refer to the gates in the column nearest to the output terminal as output gates.

Suppose that a switching gate randomizes inputs. It outputs the results in random order to conceal the correspondence between the inputs and outputs. Let $E_y(\cdot)$ and $R_y(\cdot)$ be a probabilistic encryption function and a randomization function with respect to encryption key y that satisfy $R_y(E_y(msg,s),r)=E_y(msg,s+r)$ for any message $msg\in \mathrm{MSG}_E$ for random factors s,r and for some binary operation '+'. Let $I_0:=E_y(msg_0,s_0),I_1:=E_y(msg_1,s_1)$ be inputs to a switching gate. The switching gate selects random factor r_0,r_1 and outputs $O_b:=R(I_0,r_0)=E_y(msg_0,s_0+r_0)$ and $O_{\bar{b}}:=R_y(I_1,r_1)=E_y(msg_1,s_1+r_1)$ according to $b\in_R\{0,1\}$. If E_y provides indistinguishability, that is, it is infeasible to determine $\beta\in_R\{0,1\}$ given $(msg_\beta,msg_{\bar{\beta}})$ and (I_0,I_1) , then it is also infeasible to determine b when (I_0,I_1) and (O_0,O_1) are given. Accordingly, the switching gate conceals the correspondence between its inputs and outputs. By using such switching gates for $\mathrm{PN}^{(N)}$, N randomized ciphertexts are output in random order.

To provide verifiability, the switching gate executes a zero-knowledge proof that guarantees correct application of R_y without revealing r_0, r_1, b . Such a proof can be done by combining the Chaum-Pedersen protocol [6], which proves the equality of two discrete logarithms, with the protocol used in [7], which proves two statements connected by OR. This costs about four times as much computation as the Chaum-Pedersen protocol (see section 5.2).

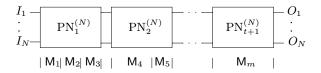


Fig. 3. Series of permutation networks. Each server is assigned some gates within a network.

To tolerate at most t honest but curious servers (i.e., passive adversaries), we construct the permutation phase with t+1 permutation networks connected in series as shown in Figure 3. As illustrated, each server is assigned some gates within a network and never works for two or more networks. With this structure, we have at least one adversary-free permutation network among them. Accordingly, those networks can yield any permutation of N inputs unknown to at most t passive adversaries.

When more than t servers agree that the permutation phase is done correctly, they proceed to the decryption phase where they decrypt the result of the first phase in collaboration.

3.2 Sketch of MiP-1

MiP-1 consists of only one phase, the randomized decryption phase, where a series of servers perform decryption and random permutation at the same time.

Similar to MiP-2, a series of t+1 permutation networks are used for constructing MiP-1. This time, however, each server is assigned to one or more column of gates within a permutation network, in contrast to MiP-2 where servers can be assigned to any gate.

Let d be the total number of columns of t+1 permutation networks. A server assigned to the gates in j-th column has an encryption key y_j and decryption key x_j . Let x be $x=x_1\oplus\cdots\oplus x_d$ and $y=y_1\otimes\cdots\otimes y_d$ for some commutative binary operations ' \oplus ' and ' \otimes '. Let \hat{y}_j denote $y_j\otimes\cdots\otimes y_d$ (so $y=\hat{y}_1$). Suppose that decryption function $D_{x_j}(\cdot)$ satisfies $D_{x_j}(E_{\hat{y}_j}(msg,s))=E_{\hat{y}_{j+1}}(msg,s)$ for any appropriate message msg and random factor s. Namely, $D_{x_j}(\cdot)$ is a function that transforms a ciphertext with \hat{y}_j into another with \hat{y}_{j+1} associated with the same plain message. (For j=d, let $D_{x_d}(E_{y_d}(msg,s))=msg$.)

Let $I_0 := E_{\hat{y}_j}(msg_0, s_0), I_1 := E_{\hat{y}_j}(msg_1, s_1)$ be inputs to a switching gate in the j-th column. The switching gate randomly chooses r_0, r_1 and applies $D_{x_j}(\cdot)$ and $R_{\hat{y}_{j+1}}(\cdot)$ to I_0 and I_1 as $O_b := R_{\hat{y}_{j+1}}(D_{x_j}(I_0), r_0), O_{\bar{b}} := R_{\hat{y}_{j+1}}(D_{x_j}(I_1), r_1)$ for $b \in_R \{0, 1\}$. It then outputs O_0, O_1 . Observe that $O_b := R_{\hat{y}_{j+1}}(D_{x_j}(I_0), r_0) = R_{\hat{y}_{j+1}}(E_{\hat{y}_{j+1}}(msg_0, s_0), r_0) = E_{\hat{y}_{j+1}}(msg_0, s_0 + r_0)$. As a result, the output of a switching gate in the (j+1)-th column is a message encrypted with key \hat{y}_{j+1} . Therefore, by applying this procedure to the last column of the series of permutation networks (the output gates only apply $D_{x_d}(\cdot)$), we obtain the plaintext as $D_{x_d}(R_{\hat{y}_d}(D_{x_{d-1}}\cdots R_{\hat{y}_2}(D_{x_1}(I_0), r_0)\cdots)) = D_{x_d\oplus\cdots\oplus x_1}(I_0) = msg_0$.

For verifiability, each gate proves in zero-knowledge that $D_{x_j}(\cdot)$ and $R_{\hat{y}_{j+1}}(\cdot)$ have been applied correctly without revealing x_j, r_0, r_1, b . It is about 6 times more costly than the Chaum-Pedersen protocol (see Section 6.2). Fixed gates in the j-th column only apply $D_{x_j}(\cdot)$ to inputs. In this case, the proof costs about half of that of the switching gates.

4 Encryption Scheme

To achieve anonymity, inputs to a mix-net must be encrypted in a non-malleable way. Although the robust threshold Cramer-Shoup cryptosystems [4,2], which are provably non-malleable, can be used for our purpose we employ the following scheme which combines El Gamal encryption and Schnorr signature for the sake of efficiency. For a rigorous discussion of the security of the scheme, please refer to [16].

Let \mathcal{G} be a discrete logarithm instance generator, which, on input security parameter 1^n , outputs (p,q,g) where p,q are large primes that satisfy p=2q+1 and g is an element of subgroup G_q of order q of multiplicative group Z_p^* . We assume the intractability of solving the discrete logarithm in G_q . Hereafter, all arithmetic operations are done in mod p unless otherwise stated.

Let $(x,y) \in Z_q \times G_q$ be a decryption and encryption key pair that satisfy $y = g^x$. A ciphertext of message $msg \in G_q$ is (M,G,c,z) where $M = msg \ y^s$, $G = g^s$, $c = \mathcal{H}(M,G,g^w)$, $z = w - cs \mod q$ for a hash function $\mathcal{H}: \{0,1\}^* \to 2^{|q|}$ and random number $s,w \in_U Z_q$. Decryption is done by first verifying whether $c = \mathcal{H}(M,G,g^zG^c)$, and $M,G \in G_q$. If it is successful, the output is $msg := M/G^x$.

The input to a mix-net is a list of (M, G)-s that successfully passes the above verifications. The list is assumed not to contain duplicated (M, G)-s.

5 Details of MiP-2

5.1 Preliminaries

Common parameters p, q, g are generated by a generator and are published to all participants in an authentic way. Threshold t is selected from 0 to $\lfloor \frac{m-1}{2} \rfloor$. As described in the overview, each server is assigned some switching gates within the same $\operatorname{PN}^{(N)}$ in a series of t+1 $\operatorname{PN}^{(N)}$. Though each server can be assigned to any (not necessarily adjacent) gates, we assume, hereafter, that all the gates of M_k are connected to the gates of M_{k+1} just for efficiency and simplicity. This limitation does not affect the descriptions given in Section 5.2. Obvious modification of descriptions in Section 5.3 will yield general description for MiP-2.

All servers cooperate to execute the key generation protocol [14,9]. As a result, decryption key x is shared into x_1, \ldots, x_m by using the (t+1, m)-threshold scheme so that server M_k has x_k privately. Corresponding public keys $y := g^x$ and all $y_k := g^{x_k}$ are published.

5.2 Task of Switching Gates

The task of a switching gate is twofold: the main operation and the proof of correctness of the operation. Server M_k executes the following for each switching gate in its charge. Let $I_i = (M_i, G_i)$ for i = 0, 1 be inputs to a switching gate.

[Main Operation]

Choose $r_0, r_1 \in_R Z_q$, and $b \in_R \{0, 1\}$. Compute

$$O_b := (\tilde{M}_b, \tilde{G}_b) = (M_0 y^{r_0}, G_0 g^{r_0}), \tag{1}$$

$$O_{\bar{b}} := (\tilde{M}_{\bar{b}}, \tilde{G}_{\bar{b}}) = (M_1 y^{r_1}, G_1 g^{r_1}).$$
 (2)

[End]

 M_k then proves in zero-knowledge that

(St-1):
$$\log_u \tilde{M}_0/M_0 = \log_q \tilde{G}_0/G_0 \wedge \log_u \tilde{M}_1/M_1 = \log_q \tilde{G}_1/G_1$$

or

(St-2):
$$\log_y \tilde{M}_0/M_1 = \log_g \tilde{G}_0/G_1 \quad \wedge \quad \log_y \tilde{M}_1/M_0 = \log_g \tilde{G}_1/G_0$$

is satisfied. Observe that (St-1) resp. (St-2) should hold for b = 0 resp. 1.

[Proving (St-1) \vee (St-2)]

SP₁**-1:** M_k selects $w_0, w_1, z_{\bar{b},0}, z_{\bar{b},1}, e_{\bar{b}} \in_R Z_q$, and computes, for i = 0, 1,

$$\begin{split} T_{b,i} &:= y^{w_i}, \\ W_{b,i} &:= g^{w_i}, \\ T_{\bar{b},i} &:= y^{z_{\bar{b},i}} (\tilde{M}_{\bar{b}\oplus i}/M_i)^{e_{\bar{b}}}, \text{ and} \\ W_{\bar{b},i} &:= g^{z_{\bar{b},i}} (\tilde{G}_{\bar{b}\oplus i}/G_i)^{e_{\bar{b}}} \quad (\oplus \text{ means XOR hereafter}). \end{split}$$

It then sends $(T_{0,0}, W_{0,0}, T_{0,1}, W_{0,1}, T_{1,0}, W_{1,0}, T_{1,1}, W_{1,1})$ to V.

SP₁**-2:** V sends $c \in_R Z_q$ to M_k .

SP₁-3: M_k computes $e_b := c - e_{\bar{b}} \mod q$ and $z_{b,i} := w_i - e_b \, r_i \mod q$ for i = 0, 1. M_k then sends $(e_0, e_1, z_{0,0}, z_{0,1}, z_{1,0}, z_{1,1})$ to V .

SP₁**-4:** V first checks $c \stackrel{?}{=} e_0 + e_1 \mod q$ and verifies that, for b = 0, 1 and i = 0, 1,

$$T_{b,i} \stackrel{?}{=} y^{z_{b,i}} (\tilde{M}_{b \oplus i}/M_i)^{e_b}, \text{ and}$$

 $W_{b,i} \stackrel{?}{=} g^{z_{b,i}} (\tilde{G}_{b \oplus i}/G_i)^{e_b}.$

If any check fails, V outputs False. True, otherwise.

[End]

Though SP_1 is described in interactive manner, a non-interactive protocol is obtained by employing Fiat-Shamir heuristics [8].

Lemma 6. SP_1 is an honest verifier zero-knowledge proof for $(St-1) \vee (St-2)$.

Proof. Correctness and zero-konwledgeness can be shown in a straightforward way. To prove soundness, we execute SP₁ twice with different challenge c_1, c_2 in step SP₁-2 by rewinding M_k . Let the answers obtained in SP₁-3 be $(e_0, e_1, z_{0,0}, z_{0,1}, z_{1,0}, z_{1,1})$ and $(e'_0, e'_1, z'_{0,0}, z'_{0,1}, z'_{1,0}, z'_{1,1})$. Since $e_0 + e_1 \neq e'_0 + e'_1$, at least either $e_0 \neq e'_0$ or $e_1 \neq e'_1$ holds. Suppose $e_0 \neq e'_0$. Then, since $y^{z_{0,0}}(\tilde{M}_0/M_0)^{e_0} = y^{z'_{0,0}}(\tilde{M}_0/M_0)^{e'_0}$ holds, $r_0 := \log_y(\tilde{M}_0/M_0) = (z_{0,0} - z'_{0,0})/(e'_0 - e_0)$ is obtained. Similarly, from $g^{z_{0,0}}(\tilde{G}_0/G_0)^{e_0} = g^{z'_{0,0}}(\tilde{G}_0/G_0)^{e'_0}$, we have $\log_g(\tilde{G}_0/G_0) = (z_{0,0} - z'_{0,0})/(e'_0 - e_0) = r_0$. Hence $r_0 = \log_y(\tilde{M}_0/M_0) = \log_g(\tilde{G}_0/G_0)$ is obtained. From $z_{0,1}, z'_{0,1}, e_0, e'_0$, we can compute r_1 that satisfies $r_1 = \log_y(\tilde{M}_1/M_1) = \log_g(\tilde{G}_1/G_1)$ in the same way. Thus, (St-1) is satisfied.

In the case of $e_1 \neq e'_1$, we can show that (St-2) holds in the same way.

Corollary 7. If honest V outputs True at the end of SP_1 , then O_0, O_1 are ciphertexts whose plaintexts are the same as those of I_0, I_1 regardless of the order.

Proof. Consider the case where (St-1) holds. Decryption of the outputs yields

$$\tilde{M}_0/\tilde{G}_0^x = M_0 y^{r_0}/(G_0 g^{r_0})^x = M_0/G_0^x$$
, and $\tilde{M}_1/\tilde{G}_1^x = M_1 y^{r_1}/(G_1 g^{r_1})^x = M_1/G_1^x$.

Thus, the results of decrypting inputs and outputs are identical. Similarly, if (St-2) holds, we have

$$\tilde{M}_0/\tilde{G}_0^x = M_1 y^{r_1}/(G_1 g^{r_1})^x = M_1/G_1^x$$
, and $\tilde{M}_1/\tilde{G}_1^x = M_0 y^{r_0}/(G_0 g^{r_0})^x = M_0/G_0^x$.

Thus, the results of decryption are identical except for the order.

If (St-1) and (St-2) hold at the same time then, from above equations, we have $M_0/G_0^{\ x}=M_1/G_1^{\ x}$. Therefore, such a case happens only if two inputs are made from the same plaintext.

Lemma 8. If the Decision Diffie-Hellman Problem is intractable, it is infeasible to determine b with probability non-negligibly better than 1/2.

Proof. Suppose that there exists a polynomial-time algorithm A_3 that, given (p,g,y,I_0,I_1,O_0,O_1) , outputs b with probability non-negligibly better than 1/2. Then, there exists a polynomial-time algorithm A_4 that, given (msg_0, msg_1) and their encryption $(I'_0,I'_1)=(E_y(msg_{\beta}),E_y(msg_{\bar{\beta}}))$ where $\beta \in_R \{0,1\}$, outputs β with the same success probability as that of A_3 .

Algorithm A_4 encrypts msg_0 , msg_1 as $(O'_0, O'_1) = (E_y(msg_0), E_y(msg_1))$, and then it inputs $(p, g, y, I'_0, I'_1, O'_0, O'_1)$ to A_3 . Finally A_4 outputs what A_3 outputs.

Observe that O'_0, O'_1 produced by A_4 has the same distribution as O_0, O_1 according to Corollary 7 and the fact that β is randomly chosen. Hence A_3 outputs the correspondence between (I'_0, I'_1) and (O'_0, O'_1) which is the same as the

correspondence between (I'_0, I'_1) and (msg_0, msg_1) . Thus, if A_3 succeeds, so does A_4 . However, if the Decision Diffie-Hellman Problem is intractable, algorithm A_4 does not exist [16]. Thus A_3 does not exist either.

Furthermore, Lemma 6 guarantees that SP_1 does not leak anything about b. Hence it is infeasible to determine b.

5.3 MiP-2 Protocol Description

[Permutation Phase]

The following steps are repeated for k = 1, ..., m.

- **P₁-1:** M_k reads the output of M_{k-1} from BB. (For k=0, M_k reads input messages.) It then performs the task of the switching gates in its charge according to the description in Section 5.2, and posts the results to BB.
- **P₁-2:** Every server verifies the output of M_k . If more than t+1 servers declare that it is faulty, M_k is disqualified. Then, the faulty result is just abandoned and the output from M_{k-1} is used for subsequent operation instead.

[End]

As a result, a list of El Gamal ciphertexts, say $\{(\tilde{M}_{\ell}, \tilde{G}_{\ell}) \mid \ell = 1, \dots, N\}$, appears on BB. Let Q_1 be the indexes of servers not disqualified in P_1 -2. The remaining servers proceed to the decryption phase, which consists of distributed decryption and zero-knowledge proofs for correctness of the decyption. The following protocol reduces computation complexity for the verifier by using the probabilistic verification [15,3] at the cost of communication complexity.

[Decryption Phase]

P₂-1: Each $M_{k \in \mathcal{Q}_1}$ chooses $w \in_R Z_q$ and computes $T_{k0} := g^w$ and $D_{k\ell} := \tilde{G}_{\ell}^{x_k}$, $T_{k\ell} := \tilde{G}_{\ell}^w$ for $\ell = 1, \ldots, N$. It then computes $e_k := \mathcal{H}(y_k, T_{k0}, D_{k1}, T_{k1}, \ldots, D_{kN}, T_{kN})$ mod q and $s_k := w - e_k x_k \mod q$. It finally sends s_k and all $D_{k\ell}$, $T_{k\ell}$ to BB.

P₂-2: For each $k \in \mathcal{Q}_1$, V computes e_k in the same way as above and checks if

$$1 = (g^{s_k} y_k^{e_k} T_{k0}^{-1})^{\gamma_0} \prod_{\ell=1}^N (\tilde{G}_\ell^{s_k} D_{k\ell}^{e_k} T_{k\ell}^{-1})^{\gamma_\ell}$$

holds for randomly picked $\gamma_1, \dots, \gamma_N \in Z_q^*$. [End]

The list of plaintexts can be obtained by computing

$$v_\ell := \tilde{M}_\ell / \prod_{k \in \mathcal{Q}_2} D_{k\ell}{}^{\lambda_{\mathcal{Q}_2,k}}$$

for all $\ell = 1, ..., N$ where Q_2 is the indices of servers whose output passes verification in P_2 -2 and $\lambda_{Q_2,k} = \sum_{\xi \in Q_2 \setminus \{k\}} \frac{\xi}{\xi - k} \mod q$.

5.4 Security

Theorem 9. MiP-2 is $(N-1, \lfloor \frac{m-1}{2} \rfloor)$ -anonymous.

Proof. We prove this theorem only against static adversaries who determine corrput servers and users in advance. First, when $t \leq \lfloor \frac{m-1}{2} \rfloor$, x is unconditionally secure against A because it is shared by (t+1,m)-VSS. Furthermore, the input ciphertexts cast by honest users give only negligibly better advantage to A in distinguishing corresponding plaintexts because the encryption scheme is semantically secure. Next, since there are t+1 permutation networks and only t adversarial servers, at least one permutation network consists of honest servers. According to Lemma 8, one can conclude that it is hard for A to determine the permutation applied by the honest permutation network. Furthermore, because the encryption scheme is non-malleabile, the plaintexts chosen by honest users are independent of those of corrupt users with overwhelming probability. So the plaintexts chosen by corrupt users have only negligibly small chance of containing some information about the plaintexts chosen by honest users. Therefore, all information except the resulting plaintexts gives A only a negligible advantage. Thus, the success probability of A differs negligibly from that of Al. This argument obviously holds for arbitrary numbers of corrupt users up to N-1.

Theorem 10. MiP-2 is $(N, \lfloor \frac{m-1}{2} \rfloor)$ -robust.

Proof. According to Lemma 6, any incorrect operation in a gate is detected with overwhelming probability, so actively deviating servers are disqualified at P_1 -2 because they are in the minority if $|\mathcal{M}_{\mathsf{A}}| \leq t \leq \lfloor \frac{m-1}{2} \rfloor$. On the other hand, honest servers will never be disqualified because they are in the majority. Similarly, any incorrect computation in the decryption phase will be detected with overwhelming probability because every server must prove correctness in zero-knowledge. From Corollary 7, one can conclude that if all proofs in the permutation phase are correct, then the output of that phase is a list of El Gamal ciphertexts that yields the same set of plaintexts as do the input ciphertexts. Since there are at most t corrupt servers, $\mathcal{Q}_2 \geq m - t \geq t + 1$ must hold. Thus M_k , $k \in \mathcal{Q}_2$ are sufficient to produce correct plaintexts because the decryption key is shared by using the (t+1,m)-threshold scheme.

The above argument holds for any input message. Furthermore, all computation in the protocols can be finished in time polynomial against input message length. Since users are polynomial bounded, their input message length is also polynomial bounded. That is, the protocol can be completed in polynomial-time regardless of the inputs. Thus, no deviation of any number of users will affect either the correctness or running time.

Theorem 11. MiP-2 is Universally Verifiable.

Proof. According to Corollary 7, if all proofs in the permutation phase are correct, then the output of that phase is a list of El Gamal ciphertexts that yield the same set of plaintexts as do the input ciphertexts. It is also true that if the proof in the decryption phase is correct, the output of that phase is a result of

decrypting the ciphertexts produced by the permutation phase. Furthermore, it is clear that all verification equations use just public variables. \Box

6 Details of MiP-1

6.1 Preliminaries

Similar to MiP-2, t+1 permutation networks are connected in sequence and every server is assigned some gates within a network. In this case, however, a server must be assigned all gates in the same column. Let d be the total number of columns of t+1 permutation networks. That is, $d=(2\log N-1)(t+1)$. Each server will be in charge of d/m columns on average. Suppose that M_k is in charge of the j-th column. M_k generates $x_j \in Z_q$ and $y_j := g^{x_j}$. It then distributes x_j to other servers with (t+1,m)-VSS in case M_k should malfunction during subsequent protocols. Let \hat{y}_j denote $y_j y_{j+1} \cdots y_d$. Encryption key y is $y := \hat{y}_1 = y_1 \cdots y_d$, which is published together with all y_j -s.

6.2 Task of Switching Gates

The task of a switching gate in the j-th column is to decrypt the inputs with x_j , randomize the result, and output them in random order. Such a technique was first used in [13].

[Main Operation]

Select $r_0, r_1 \in_R Z_q$, $b \in_R \{0, 1\}$ and compute

$$O_b := (\tilde{M}_b, \tilde{G}_b) = (M_0 G_0^{-x_j} \hat{y}_{i+1}^{r_0}, G_0 g^{r_0}), \text{ and}$$
 (3)

$$O_{\bar{b}} := (\tilde{M}_{\bar{b}}, \tilde{G}_{\bar{b}}) = (M_1 G_1^{-x_j} \hat{y}_{j+1}^{r_1}, G_1 g^{r_1}). \tag{4}$$

[End]

To guarantee that the main operation step is done correctly, it proves in zero-knowledge that the input and output satisfy

(St-3):
$$\tilde{M}_0/M_0 = G_0^{-x_j} \hat{y}_{j+1}^{r_0} \wedge \tilde{G}_0/G_0 = g^{r_0} \wedge \tilde{M}_1/M_1 = G_1^{-x_j} \hat{y}_{j+1}^{r_1} \wedge \tilde{G}_1/G_1 = g^{r_1}$$

or

(St-4):
$$\tilde{M}_1/M_0 = G_0^{-x_j} \hat{y}_{j+1}^{r_0} \wedge \tilde{G}_1/G_0 = g^{r_0} \wedge \tilde{M}_0/M_1 = G_1^{-x_j} \hat{y}_{j+1}^{r_1} \wedge \tilde{G}_0/G_1 = g^{r_1}$$

for some r_0, r_1 and for x_j that satisfies $y_j = g^{x_j}$.

[Proof of (St-3) \vee (St-4)]

SP₂**-1:** M_k randomly selects $w_0, w_1, v, z_{\bar{b},0}, z_{\bar{b},1}, s_{\bar{b}}, e_{\bar{b}}$ from Z_q . It then computes, for i=0,1,

$$\begin{split} T_{b,i} &:= G_i^{-v} \, \hat{y}_{j+1}^{w_i}, \\ W_{b,i} &:= g^{w_i}, \\ V_b &:= g^v, \\ T_{\bar{b},i} &:= G_i^{-s_{\bar{b}}} \, \hat{y}_{j+1}^{z_{\bar{b},i}} (\tilde{M}_{\bar{b}\oplus i}/M_i)^{e_{\bar{b}}}, \\ W_{\bar{b},i} &:= g^{z_{\bar{b},i}} (\tilde{G}_{\bar{b}\oplus i}/G_i)^{e_{\bar{b}}} \\ V_{\bar{b}} &:= g^{s_{\bar{b}}} \, y_j^{e_{\bar{b}}}, \end{split}$$

and sends $(T_{0,0}, W_{0,0}, T_{0,1}, W_{0,1}, V_0, T_{1,0}, W_{1,0}, T_{1,1}, W_{1,1}, V_1)$ to V.

 \mathbf{SP}_2 -2: V sends $c \in_R Z_q$ to M_k .

 \mathbf{SP}_2 -3: M_k computes

$$e_b := c - e_{\bar{b}} \mod q,$$

 $s_b := v - e_b x_j \mod q, \text{ and}$
 $z_{b,i} := w_i - e_b r_i \mod q \text{ for } i = 0, 1.$

It then sends $(e_0, e_1, z_{0,0}, z_{0,1}, s_0, z_{1,0}, z_{1,1}, s_1)$ to V.

SP₂**-4:** V first checks $e_0 + e_1 \stackrel{?}{=} c \pmod{q}$, then verifies that

$$T_{b,i} \stackrel{?}{=} G_i^{-s_b} \hat{y}_{j+1}^{z_{b,i}} (\tilde{M}_{b \oplus i}/M_i)^{e_b},$$

$$W_{b,i} \stackrel{?}{=} g^{z_{b,i}} (\tilde{G}_{b \oplus i}/G_i)^{e_b}$$

$$V_b \stackrel{?}{=} g^{s_b} y_j^{e_b}$$

for b = 0, 1 and i = 0, 1. If it holds, V outputs True, or False otherwise.

[End]

Lemma 12. SP_2 is an honest verifier zero-knowledge proof for $(St-3) \vee (St-4)$.

Corollary 13. If honest V outputs True at the end of SP_2 , then decryption of inputs with key $x_j + \cdots + x_d \mod q$ and decryption of outputs with key $x_{j+1} + \cdots + x_d \mod q$ results in the same list of plaintexts without regard to the order.

The proof follows that for Collorary 7.

Lemma 14. If the Decision Diffie-Hellman Problem is intractable, it is infeasible to determine b with probability non-negligibly better than 1/2.

Proof. (Sketch) Suppose that there exists a polynomial-time algorithm A_5 that, given $(\tilde{M}_0, \tilde{G}_0)$, $(\tilde{M}_1, \tilde{G}_1)$, (M_0, G_0) , (M_1, G_1) and $y_j, \ldots, y_d, x_{j+1}, \ldots, x_d$ that satisfy Relation 3 and 4, outputs correct b with probability non-legligibly better

than 1/2. We can then show that there exists algorithm A_6 that distinguishes the correspondence between a pair of El Gamal ciphertexts for encryption key y_j and a pair of messages with probability non-neglegibly better than 1/2. Since such A_6 does not exist if the Decision Diffie-Hellman Problem is intractable, neither does A_5 . Moreover, Lemma 12 states that the proof is zero-knowledge. Thus, it is hard to guess b with probability non-negligibly better than 1/2. \square

6.3 Task of Fixed Gates

The fixed gates in the j-th column simply perform decryption as follows.

[Main Operation]

$$(\tilde{M}_0, \tilde{G}_0) := (M_0 G_0^{-x_j}, G_0)$$

 $(\tilde{M}_1, \tilde{G}_1) := (M_1 G_1^{-x_j}, G_1)$

[End]

To guarantee that the above process is done correctly, a server proves in zero-knowledge that the inputs and outputs satisfy

$$\log_{G_0} \tilde{M}_0 / M_0 = \log_{G_1} \tilde{M}_1 / M_1 = -\log_q y_j.$$

The precise protocol can be easily derived from the Chaum-Pedersen protocol and so is omitted.

6.4 Task of Output Gates

Each output gate performs decryption and random permutation (without randomization) as follows.

[Main Operation]

For $b \in \mathbb{R} \{0,1\}$, compute

$$\tilde{M}_b := M_0 G_0^{-x_d}$$
, and $\tilde{M}_{\bar{b}} := M_1 G_1^{-x_d}$.

[End]

The proof for the above process involves showing that the inputs and outputs satisfy

(St-5):
$$\log_{G_0} \tilde{M}_0/M_0 = \log_{G_1} \tilde{M}_1/M_1 = -\log_q y_d$$

or

(St-6):
$$\log_{G_1} \tilde{M}_0/M_1 = \log_{G_0} \tilde{M}_1/M_0 = -\log_a y_d$$
.

The precise protocol can be obtained from SP_2 by removing the factors related to r_0 and r_1 .

6.5 MiP-1 Protocol Description

[Randomized Decryption Phase]

P₃-1: The following is repeated for k = 1, ..., m: M_k verifies all proofs issued by previous servers. If successful, M_k takes the output of M_{k-1} as its input and performs the task of each gate in its charge. If it finds M_{bad} faulty, it declares so and goes into the disqualification procedure (see below). Then M_k performs his task by taking the result of public decryption done in the disqualification procedure as its input.

Disqualification Procedure: Every server verifies the proofs issued by M_{bad} . If more than t+1 servers conclude that M_{bad} is faulty, all servers from M_{bad} to M_{k-1} are disqualified and all partial decryption keys belonging to those servers are published by VSS reconstruction. Partial decryption with those keys is then done in public in the place of the disqualified servers.

 P_3 -2: V verifies all outputs appearing on the BB. If V finds wrong intermediate results, it invokes the disqualification procedure to obtain the decryption keys of the faulty servers.

[End]

Alhough servers use BB in above description, M_k can directly send all results to M_{k+1} together with all inputs received from M_{k-1} in an authentic and non-repudiable way.

Similar to MiP-2, the following theorem holds.

Theorem 15. MiP-1 is $(N-1, \lfloor \frac{m-1}{2} \rfloor)$ -anonymous, $(N, \lfloor \frac{m-1}{2} \rfloor)$ -robust and Universally Verifiable.

7 Efficiency

Computational efficiency is estimated based on the number of modular exponentiations in $\mod p$. Since the servers have different amounts of tasks in both of our schemes, we discuss average cases. In the following, we count the cost of computing a single-base modular exponentiation as 1. By assuming the use of the simple table lookup method, double, triple, and quadruple-base modular exponentiation are counted as 1.2, 1.25, and 1.31 respectively. Table 1 shows the computational cost for the main operations, proofs, and verification at each gate.

MiP-2 needs further costs in the decryption phase: each server pays 1+N for decryption, N for proving correct decryption, and (m-1)(1+1.2N) for verifying the proof issued by other servers. The number of switching gates in MiP-2 is $(N\log_2 N - N + 1)(t+1)$. MiP-1 uses $(N\log_2 N - N + 1)(t+1) - (N/2 - 1)$ switching gates, N/2 - 1 fixed gates, and N/2 - 1 output gates.

According to our estimation, MiP-1 is faster than the DL-based scheme of [12] for $N < 2^{40}$ with $\kappa = 80$ and m = 5, which is the same setting as used in [1].

1			main operation	proof	verify
ı	MiP-2	switching gate	4	9.2	10.4
ı	MiP-1	switching gate	4.4	9.52	10.24
ı	•	fixed gate	2	2	2.5
ı		output gate	2	4.5	5

Table 1. Computational cost for a gate.

With the same parameters, MiP-2 is faster for $N < 2^{25}$. For instance, with $N = 2^{12}$ and $N = 2^7$, MiP-1 is 3.6 and 6.7 times faster than [12] respectively. In the same setting, MiP-2 is 2.2 and 4.0 times faster than [12].

Efficiency can be increased by employing pre-computation for all fixed-base modular exponentiation with random exponents. Furthermore, the probabilistic verification used in P_2 -2 improves efficiency if it is used in P_3 -4 and P_3 -4.

8 Concluding Remarks

The major difference between the constructions, MiP-1 and MiP-2, is that each server in MiP-2 verifies all results by itself while servers in MiP-1 do not unless triggered by others. Accordingly, MiP-2 will be used in applications where the servers eventually use the output for further computation. On the other hand, MiP-1 suits applications where mix servers work as a gateway and the resulting plaintexts are used by other entities.

Another considerable difference is their recovery from disruption. If a deviating server is detected in MiP-2, other severs simply skip his results while servers in MiP-1 must cooperate to recover decryption keys and perform decryption in public each time a deviating server is detected. Accordingly, MiP-1 suffers more round/communication/computation complexity for recovery than MiP-2. However, this disadvantage can be eliminated as follows: Suppose a server, which hosts gates in the j-th column, finds faults among the output of gates in the (j-1)-th column. The server then takes the output from the (j-2)-th column as its input and uses $y_{j-1}\hat{y}_{j+1}$ for randomization (in Equation 3 and 4) instead of \hat{y}_{j+1} . This, intuitively, corresponds to putting off the decryption process for j-th column to the end of the Mix-net. Corresponding proofs can be generated appropriately. After M_m outputs the result, V requests all servers to reveal pieces of x_{j-1} so that the verifier can execute the skipped decryption process by reconstructing x_{j-1} .

Acknowledgements

The author wishes to thank Kazue Sako and Tatusaki Okamoto for their helpful suggestions on the definitions in Section 2. Constructive discussions with A. Fujioka, F. Hoshino, and M. Michels are also acknowledged. Finally the author

thank the anonymous refrees for comments which improved presentation of the paper.

References

- M. Abe. Universally verifiable mix-net with verification work independent of the number of mix-servers. In Advances in Cryptology — EUROCRYPT '98, LNCS 1403, pages 437–447. Springer-Verlag, 1998. 258, 260, 271
- M. Abe. Robust threshold Cramer-Shoup cryptosystem. Proc. of the 1999 Symposium on Cryptography and Information Security, T1-1.3, 1999. (in Japanese). 263
- M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In Advances in Cryptology EUROCRYPT '98, LNCS 1403, pages 236–250. Springer-Verlag, 1998. 266
- R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In Advances in Cryptology — EUROCRYPT '99, LNCS 1592, pages 90–106. Springer-Verlag, 1999. 263
- 5. D. L. Chaum. Untraceable electronic mail, return address, and digital pseudonyms. Communications of the ACM, 24:84–88, 1981. 258
- D. L. Chaum and T. P. Pedersen. Wallet databases with observers. In Advances in Cryptology — CRYPTO '92, LNCS 740, pages 89–105. Springer-Verlag, 1993.
 261
- R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Advances in Cryptology — CRYPTO '94, LNCS 839, pages 174–187. Springer-Verlag, 1994. 261
- 8. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology CRYPTO '86*, LNCS 263, pages 186–199. Springer-Verlag, 1986. 264
- R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In Advances in Cryptology EURO-CRYPT '99, LNCS 1592, pages 259–310. Springer-Verlag, 1999. 263
- M. Jakobsson. A practical mix. In Advances in Cryptology EUROCRYPT '98, LNCS 1403, pages 448–461. Springer-Verlag, 1998.
- 11. M. Jakobsson. Flash mixing. *PODC99*, 1999. 258
- W. Ogata, K. Kurosawa, K. Sako, and K. Takatani. Fault tolerant anonymous channel. In *ICICS98*, LNCS 1334, pages 440–444. Springer-Verlag, 1998. 258, 259, 260, 271, 272
- C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In Advances in Cryptology — EUROCRYPT '93, LNCS 765, pages 248–259. Springer-Verlag, 1994. 258, 268
- T. P. Pedersen. A threshold cryptosystem without a trusted party. In Advances in Cryptology — EUROCRYPT '91, pages 522–526. Springer-Verlag, 1991. 263
- 15. K. Sako and J. Kilian. Receipt-free mix-type voting scheme a practical solution to the implementation of a voting booth —. In *Advances in Cryptology EUROCRYPT '95*, LNCS 921, pages 393–403. Springer-Verlag, 1995. 258, 266
- Y. Tsiounis and M. Yung. On the security of El Gamal based encryption. In First International Workshop on Practice and Theory in Public Key Cryptography – PKC '98, LNCS 1431, pages 117–134. Springer-Verlag, 1998. 263, 266
- A. Waksman. A permutation network. Journal of the Association for Computing Machinery, 15(1):159–163, January 1968.

Secure Communication in an Unknown Network Using Certificates*

Mike Burmester 1** and Yvo Desmedt 2,1***

¹ Information Security Group, Department of Mathematics, Royal Holloway – University of London, Egham, Surrey TW20 OEX, UK, m.burmester@rhbnc.ac.uk, http://hp.ma.rhbnc.ac.uk/~uhah205/

Department of Computer Science, Florida State University, Tallahassee Florida FL 32306-4530, USA, desmedt@cs.fsu.edu, http://www.cs.fsu.edu/~desmedt

Abstract. We consider the problem of secure communication in a network with malicious (Byzantine) faults for which the trust graph, with vertices the processors and edges corresponding to certified public keys, is not known except possibly to the adversary. This scenario occurs in several models, as for example in survivability models in which the certifying authorities may be corrupted, or in networks which are being constructed in a decentralized way. We present a protocol that allows secure communication in this case, provided the trust graph is sufficiently connected.

1 Introduction

Secure communication in an open and dynamic network in the presence of a malicious adversary can only be achieved when the messages are authenticated. For this purpose we use authentication channels. There are several ways to establish such channels. For example, we can use dedicated communication lines in the network. Alternatively, shared secret keys or public keys can be used. The graph with vertices the processors in the network and edges the authentication channels is called a trust graph [5]. If the sender is connected to the receiver by an edge in this graph then the messages can be authenticated through the corresponding channel. Otherwise we may use authentication paths through intermediary processors in the trust graph [20,5].

^{*} Research supported by DARPA F30602-97-1-0205. However the views and conclusions contained in this paper are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advance Research Projects Agency (DARPA), the Air Force, of the US Government.

 $^{^{\}star\star}$ Part of this research was done while visiting the University of Wisconsin – Milwaukee.

^{***} This research was done while the author was at the University of Wisconsin – Milwaukee.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 274–287, 1999. © Springer-Verlag Berlin Heidelberg 1999

The interplay of network connectivity and communication security has been studied extensively in recent years (see e.g. [7,14,13,6,1,10,22]). Dolev [7] and Dolev-Dwork-Waarts-Yung [8] have shown that if the number of malicious (Byzantine) faulty processors (nodes) is bounded by u then secure communication can only be achieved if the network is at least (2u+1) connected. Even if the faults are not malicious, for reliable communication the network must be at least (u+1) connected.

In this paper we deal with the case when the trust graph is not known except possibly to the adversary. This scenario was first discussed in [5]. It is obvious that when there are no malicious faults, we can achieve secure communication if there are at least (2u+1) vertex disjoint paths which connect the sender and the receiver, where u is the number of faulty processors (first use standard algorithms [2] to find the trust graph). Recently this result has been extended to include the case when the faults are malicious (Byzantine), provided the trust graph is known to at least one non-faulty processor and the graph is (2u+1) connected [4]. However the case when the trust graph is not known to any (non-faulty) processor and there are malicious faults has not been investigated. In particular, no efficient algorithm for constructing the trust graph in this case has been presented so far.

In this paper we focus on the case when the authentication in the trust graph $G^* = (V^*, E^*)$ is based on public keys (using signatures), with edges $(v, w) \in E^*$ corresponding to certificates in which w certifies the public key of v (by signing it). We consider the problem of secure communication when the public key of the sender is not certified by the receiver [19], and when the structure of the trust graph is not known to the sender or the receiver. We describe an algorithm for this setting which makes it possible for the sender to compute a good approximation of the trust graph in polynomial time if the vertex-connectivity of the trust graph is at least $\lfloor 5u/2+1 \rfloor$, where u is an upper bound on the number of faulty processors.

Related Work and Motivation

This problem is an extension of the classic Byzantine generals problem [18,16,8] and is related to dependable computation. Authentication in open networks was considered by Beth-Borcherding-Klein [3] and Maurer [17]. Reiter-Stubblebine [20] consider trust-paths for authentication, and similarly Burmester-Desmedt-Kabatianski [5], but they use a slightly different model.

Goldreich-Goldwasser-Linial [12], Franklin-Yung [11] and Franklin-Wright [10] have studied broadcast (multi-recipient) models. Franklin-Wright have shown that if the number of Byzantine faults is bounded by u then we have secure communication in polynomial time if the number of disjoint "broadcast lines" is greater than $\lceil 3u/2 \rceil$. This bound has been recently lowered to greater than u [22].

In our scenario the sender has only local information about the trust graph, and in order to communicate with other processors, must find appropriate communication paths through possibly corrupted processors. This situation occurs in survivability models for which the certifying authorities may be corrupted and only local data is reliable. In this case the certifying authorities can provide erroneous (made-up) keys in order to decrypt private messages and to sign fraudulent messages [19]. This model is used by Zimmermann [23], Burmester-Desmedt-Kabatianskii [5], Rivest-Lampson [21] and Reiter-Stubblebine [20]. We use a similar model, only for us the trust graph is not known (in [23] and [21] the trust graph is known).

2 Model and Primitives

A communication system consists, essentially, of several linked processors, such as servers, programs, hardware units etc. A basic requirement is that the system should be reliable and dependable (robust). Reliability usually deals with faults which follow a random pattern, e.g., accidental faults. These faults follow predictable patterns and are usually independent of each other at their origin [5]. They can be controlled by using redundancy (replication). Dependability deals with Byzantine (malicious) faults which are more difficult to deal with. The usual scenario is for an adversary to control all the faulty processors, according to some plan which may exploit the possible weaknesses of the system. The adversary has at least as much power and knowledge of the state of the system as the non-faulty processors (excluding the secret keys), and possibly more. In our case, for example, the adversary may know the structure of the trust graph, whereas the receiver will not. The adversary may try to use such information and forward misleading messages to non-faulty processors in an attempt to make the system fail, as in the case of the boqus paths attack [5].

Modeling a scenario in which the adversary is malicious should allow for a dynamic topology in which changes in the system may take place without the (non-faulty) processors being aware of it. It should allow for the most general type of processor which could represent a simple gate, a software package, or a powerful computer. Also, the model should describe the structure of the system at the appropriate level of abstraction: it must distinguish those aspects which are relevant to the computation and abstract out those aspects which are not essential.

2.1 The Trust Graph

The trust graph $G^* = (V^*, E^*)$ is a directed graph with vertices the processors of the network and edges the authentication channels. These channels can be quite general. They can be physically secure (dedicated) communication lines. Alternatively, they can correspond to conventional authentication channels with shared secret keys, or to authentication channels with public keys. We are not interested in how these channels are implemented, or how the underlying (real) communication network is used, and we do not make any specific requirements other than that these channels are reliable and that we have synchrony (delays are bounded). This issue will be discussed in Section 6.1.

We are concerned with the secure communication between the (non-faulty) processors of the trust graph. Since this can be achieved when this graph is known [5], or when it is completely connected, we focus on the case when its structure is not known and when it has a weaker connectivity. Our goal is to find a polynomial time algorithm to construct the trust graph, or at least a good approximation of it. We summarize our basic requirements for this model below.

General Assumptions

- The authentication channels are reliable and we have synchrony (delays are bounded).
- All processors including the adversary are polynomially bounded (for unconditional security we allow the adversary to have unlimited computer power).
- The number of faulty processors is bounded by u, and the trust graph is |5u/2+1| vertex-connected [9].
- Every processor has a unique identifying label.

In this paper we focus on the particular case when the authentication in the trust graph $G^* = (V^*, E^*)$ is based on public keys, with signatures. Each edge $(v, w) \in E^*$ is labeled by a certificate $c_{vw} = (v, w, k_v, k_w, sign_{k_w}(v, k_v))$, where k_v, k_w are the public keys¹ of v, w, and $sign_{k_w}(v, k_v)$ is the signature of w with key k_w on (v, k_v) .

Observe that a processor v can be identified by its public key k_v . This does not restrict the generality of our approach, since we are assuming that the processors have unique identifying labels. When there is no ambiguity we may identify v with k_v . It is important however to note that in our model there is a clear distinction between the label of a vertex v and its public key k_v . Indeed in our scenario a processor b wishes to communicate with a possibly known processor v and v has not certified the public key of v. A variant scenario is when v wishes to communicate with a processor v and v does not know v public key. This issue will be discussed in more detail in Section 6.2. We just mention here that if a trust graph (or a good approximation of it) has been constructed with vertices labeled by public keys then it is easy to find the "true" labels of the vertices, provided the trust graph is sufficiently connected (by querying each "public key" for its label).

2.2 A Good Approximation of the Trust Graph

The vertices of the trust graph correspond to faulty and non-faulty processors. Similarly, the edges of the trust graph correspond to faulty and non-faulty channels. Faulty processors (channels) are real processors (channels) which are under the control of the adversary. It may not be possible in the general case for a non-faulty processor to construct the true trust graph, because its faulty neighbors under the control of the adversary can always lie about their neighbors (and

 $^{^{1}}$ We shall assume that the length of the public keys is superpolynomial in u.

the neighbors of their neighbors, \ldots), e.g. by claiming or disclaiming incident edges. That is, faulty processors can make up non-existing processors and channels to prevent non-faulty processors from finding the trust graph. We call these processors and channels, fake. Fake processors (channels) do not belong to the trust graph. They correspond to vertices (edges) of a virtual graph which is an extension of the trust graph.

We say that a graph G'=(V',E') is a good approximation of the trust graph $G^*=(V^*,E^*)$ if $V'=V^*$ and $E'\subseteq E^*$, where the edges in $E^*\setminus E'$ are directed into a faulty vertex. Good approximation graphs are adequate for secure communication, provided our assumptions in Section 2.1 are satisfied. Observe that the adversary can reduce the connectivity of the trust graph from $\lfloor 5u/2+1 \rfloor$ to (2u+1) by removing (disclaiming) $\lfloor u/2 \rfloor$ edges which connect u faulty processors in pairs. For example, in Figure 1 two faulty vertices f_1, f_2 can reduce the connectivity between the two vertices r and b from 5 to 4 in such a way that they can still control 2 out of 4 vertex-disjoint paths linking r to b (if f_1 disclaims the edge (f_1, f_2)). This implies that if a message is sent through these paths, then a majority vote on the received communication may not be decisive. Of course there may be other sets of 4 vertex-disjoint paths linking r, b in which f_1, f_2 have a minority vote, but it may be hard to find these in the general case.

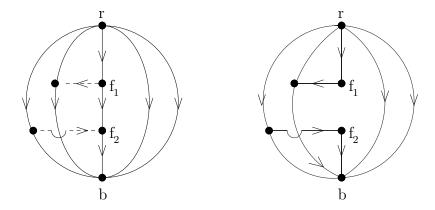


Fig. 1. Two faulty vertices f_1 , f_2 reduce the connectivity between r and b from 5 to 4 in such a way that they control 2 out of 4 vertex-disjoint paths from r to b

Any further removals by the adversary will however only reduce the number of faulty or fake paths, and will not affect the (u+1) vertex-disjoint authentication paths which are not faulty.

2.3 Virtual Paths

A path $\pi = (b_1, b_2, \dots, b_n)$ is virtual if every processor $b_{\ell}, \ell > 1$, in π has certified (signed) with its public key $k_{b_{\ell}}$, its parent $b_{\ell-1}$ and the key $k_{b_{\ell}-1}$. The description of a virtual path π must include all the certificates $c_{b_{\ell-1}b_{\ell}} =$ $(b_{\ell-1}, b_{\ell}, k_{b_{\ell-1}}, k_{b_{\ell}}, sign_{k_{b_{\ell}}}(b_{\ell-1}, k_{b_{\ell-1}})), \ \ell = 2, \ldots, n.$ To authenticate a message m through π , each processor b_{ℓ} in turn signs (m,π) and forwards this signature, together with the signatures of its ancestors to its descendants. The authentication is initialized by b_1 which forwards $\operatorname{sign}_{b_1,\pi}(m) := (m,\pi,\operatorname{sign}_{b_1}(m,\pi))$ to b_2 . Each b_{ℓ} in π , $\ell = 2, \ldots, n-1$, on receiving $\operatorname{sign}_{b_{\ell-1},\pi}(m)$ checks it for correctness, i.e., that the keys certify what they are supposed to and that the signatures are valid. If not, the message is not forwarded. Otherwise, it appends to the message its signature $\operatorname{sign}_{b_{\ell}}(m,\pi)$ and then forwards the resulting list $\operatorname{sign}_{b_{\ell},\pi}(m)$ to $b_{\ell+1}$. The message m is authenticated through the virtual path π when b_n receives a valid $\operatorname{sign}_{b_{n-1},\pi}(m)$. A path whose processors belong to the trust graph, *i.e.*, are not fake, is an authentication path. A message authenticated through such a path with no faulty processors is authentic. However if some processors are faulty then the message (e.g., a certificate) may be a forgery.

If a path π has faulty or fake processors then it is not certain if the message will ever reach b_n . A faulty processor b_ℓ can claim to its descendants that a message has been authentication by its ancestors $b'_{\ell-1}, \ldots$, but b_n cannot be certain, (i) that the message, if any, has been substituted and, (ii) that some of the processors $b'_{\ell-i}$ and edges $(b'_{\ell-i}, b'_{\ell-i+1})$ on the claimed path are not fake (not in the trust graph).

Definition. A fake path is a virtual path some of whose vertices are fake.

The following result will be needed later.

Lemma 1. Let $\pi = (b_1, b_2, \dots, b_n)$ be a virtual path. If the vertex b_n is not faulty and π is fake, then at least one ancestor b_ℓ of b_n in π must be faulty.

Proof. The public key of a fake processor will only be certified by a faulty processor.

Virtual paths can be used for secure communication if no more than u processors are faulty and if (2u+1) vertex-disjoint such paths connect the sender to the receiver. By the Lemma, if the sender and receiver are not faulty, at least (u+1) of these paths are non-faulty authentication paths. A majority vote on the received communication can then be used.

Another way to communicate in a network which is under the control of a malicious adversary is by flooding.

2.4 Flooding

Flooding [2] is a broadcasting method in which a processor x sends a message to its neighbors, which then relay it to their neighbors, and so on, until the message reaches all the processors in the trust graph (our connectivity assumption

guarantees this). To limit the number of transmissions, a processor does not relay back a message to the processor which sent it. Also, transmissions are not repeated (by using sequence numbers). If the the adversary does not make any fake processors or channels (e.g., if the faults are not malicious) then the total number of transmissions for one query through the trust graph is bounded by $2|E^*|$, where E^* is the edge set of the trust graph [2].

In our case the faulty processors are under the control of the adversary and will make fake processors and channels, and furthermore they may try to jam the system by claiming to have a large number of (mostly fake) neighbors. To prevent this we introduce Round Robin Flooding (round robin was used before, in a different context, to solve a security problem [15]). In this, each processor x allocates "equal-time" to all its edges. For convenience we take the delay-time of the authentication channels (the edges) in the trust graph G^* to be bounded by 1. Then x will allocate to each of its incoming edges time bounded by deg(x), the degree of x in G^* . This means that the time taken for a query of a non-faulty processor to reach any other processor in G^* is bounded by n^2 , where $n = |V^*|$, provided G^* is (u+1) vertex-connected, with u an upper bound on the number of faulty processors. Since we only use Round Robin Flooding, from now on we shall refer to this simply as flooding.

3 Secure Communication with Byzantine Faults

We can formulate our problem of secure communication in terms of communication networks. These networks can be represented by graphs G=(V,E) in which communication is possible only through the edges of G (we assume that these are reliable). In our case up to u vertices in G may be faulty and under the control of the adversary. Communication through these may be corrupted. In particular, a faulty vertex can lie about its neighbors. If the graph has sufficient connectivity then secure communication can be achieved through (2u+1) vertex-disjoint communication paths, since the adversary can only occupy less than half of these. However in our case the structure of G is not known. The problem is to find an efficient algorithm to construct G, or at least a good approximation of G, in the case when the adversary can control up to u vertices. There are two different ways in which this problem can be stated.

 $\mathbf{CN1}$ - Constructing a communication network with up to u faulty vertices and $Adversary_u$.

Instance: A directed graph G = (V, E), $b \in V$, the set N_b of neighbors of b in V, the set E_b of edges in E incident to b, the $Adversary_u$ which can control up to u vertices in G.

Question: Can a good approximation of G be constructed given as input only $b \in V$, N_b and E_b , in the presence of $Adversary_u$.

In this problem, the vertex b only knows its neighbors in N_b and the corresponding edges E_b of the communication graph, and does not have access to the program of $Adversary_u$. It can find out information about G by communicating

through its neighbors (or by guessing). We assume that the adversary has a *fixed* program. This restriction is removed in the next problem.

CN2 – Constructing a communication network with up to u faulty vertices and any adversary.

Instance: A directed graph G = (V, E), $b \in V$, the set N_b of neighbors of b in V, the set E_b of edges in E incident to b.

Question: Can a good approximation of G be constructed given as input only $b \in V$, N_b and E_b , in the presence of any adversary which can corrupt up to u vertices in G.

This problem addresses malicious faults in a general way. It gives more power to the adversary who can change dynamically her program, whereas the non-faulty processors are bound by their programs.

We will show that both problems can be solved in polynomial time. We discuss the first one in the following section. In Section 4 we deal with the second problem.

3.1 Problem CN1 – A Simplistic Solution

Suppose that vertex b wants to construct a good approximation of the trust graph $G^* = (V^*, E^*)$ by querying all its neighbors in G^* , the neighbors of the neighbors, etc, for a signed list of the labels (the certificates) of their incoming edges. The query is flooded, and the neighbor list of vertex x is $L_x := (x, E_x^*, \operatorname{sign}_{k_x}(E_x^*))$, where E_x^* is the list of labels of the incoming edges of x in E^* . Lists are only forwarded or accepted if they are correct (the signature in L_x must authenticate the labels with the key k_x of x, and the labels must be valid). Let n be the order of the graph G and let n^c be an upper bound on the complexity of $Adversary_u$. Vertex b will receive at most,

$$(n-u) + un^c \le n^{c+1}$$

edge lists. This is a first approximation G'' = (V'', E'') of the trust graph G^* . By our connectivity requirements on G^* , G'' must contain a good approximation of G^* .

The next stage is to remove from G'' all fake processors. Let $x \neq b$ be any vertex in V'', and c(x,b) be the connectivity from x to b in G'', that is the maximum number of vertex-disjoint paths connecting x to b in G''. If the vertex x is fake then $c(x,b) \leq u$ by Lemma 1. On the other hand if x is not fake then $c(x,b) \geq u+1$, by our connectivity assumption. Checking this connectivity can be done by using a Max Flow algorithm² with complexity $O(|V''|^{1/2} \cdot |E''|)$ (Dinic's algorithm [9]). The complexity of finding all the non-fake vertices is therefore $O(n^{5/2(c+1)})$. Let V' be the set of vertices in V'' which belong the trust graph (the non-fake vertices in V'') and let E' the set of edges in E''

² Crucial to our argument is the fact that the adversary cannot make fake labels for edges (v, w) with non-faulty vertices.

which belong the trust graph (the non-fake edges in E''). Then G' = (V', E') a good approximation of the trust graph. It follows that vertex b will get a good approximation of the trust graph in polynomial time by using this procedure. Of course we must assume that the signature scheme will remain secure in this adverserial scenario.

This construction is not really satisfactory because its complexity is a function of the complexity of $Adversary_u$. In particular, it requires that non-faulty vertices work harder than faulty vertices.

4 Problem CN2 – the General Case

4.1 Discussion

The main problem with the construction in Section 3.1 is that there is no halting strategy. The construction goes on until $Adversary_u$ is exhausted, which of course is too late. The trust graph will be completed long before the construction has ended, and what is constructed includes mainly fake vertices and edges. What we need is some means to recognize this.

There are also other problems with this construction. A neighbor list of a faulty vertex can be enormous, consisting mainly (or entirely) of endless lists of fake vertices and edges. If such a vertex were given equal time to a non-faulty vertex, this would allow the adversary to take control over most of the construction. The easiest way to sort out this problem is to share time equally, and ask each vertex to send only one edge label at-a-time.

Finally, by having no bound on the order of the graph, the description of some of the vertices may be much longer than needed. We shall assume that the description of the vertices of the trust graph is short. The problem with the description of fake vertices is dealt with by using a subprotocol (Round Robin) packet_flood in which the sender sends one packet at-a-time. The receiver will use these only after an EOT (end of transfer) has been received.

4.2 An Informal Approach

The first part of our construction is similar to the previous one, modified to take into account our remarks in the previous section. Vertex b floods a query in the trust graph G^* to all other vertices for signed labels of their incoming edges, one-at-a-time. At some stage b will start receiving signed labels of edges (v,w) from which it can begin to build a graph G''. After some time, some of the vertices will have completed their lists. These vertices are labeled "replied". The others, in the process of replying, are labeled "replying". Eventually some of the vertices will be linked to b. These are labeled "linked". The others are "not_yet_linked".

Suppose that the graph G'' under construction has reached the point when $V' \subseteq V''$, *i.e.*, the vertices in the good approximation graph G' have all been found. Then the vertices v'' in G'' which are still sending new vertices v''', *i.e.*,

not yet found, must be under the control of the adversary. These v'' vertices are either faulty or fake. Let G''_{aux} be the graph obtained from G'' by adding one new vertex v_{aux} and new edges connecting v_{aux} to all the vertices in G'' labeled replying. It is easy to see that $V' \subseteq V''$ if and only if $c(v_{aux}, b) \leq u$, Indeed there are at most u faulty vertices. We will use this test for our halting procedure.

Halting routine: $is_graph(*)$

Argument: $G'' = (V'', E''), b \in V'',$ a list of vertices $x_i \in V''$ labeled replying. **Value:** satisfactory if $c(v_{aux}, b) \leq u$, else not_satisfactory.

Observe however, that this does not guarantee that G'' contains a good approximation graph. We only know that all vertices in G' have been found. Processor b will now ask all vertices v labeled "replying" to give, a new incoming edge, one at-a-time, as before. If (w, v) is such an edge, but $w \notin V''$, then b stops asking v for new edges.

Find missing edges routine: missing_edges(*)

Argument: $G'' = (V'', E''), b \in V''$, a list of vertices $x_i \in V''$ labeled *replying*. **Value:** a graph G''', containing a good approximation graph.

Now suppose that processor b has constructed a first approximation of the trust graph G^* , that is a graph G'' = (V'', E'') which contains a good approximation G' of G^* . G' will also contain fake vertices and edges. Fake vertices x can be traced as in Section 3.1, because $c(x, b) \leq u$. Similarly fake edges can be traced. Discarding these from G'' will give us a good approximation of the trust graph.

Cleaning up routine: clean_up (*)

Argument: G'' = (V'', E''), a first approximation graph.

Value: G', a good approximation graph.

4.3 The Protocol

Vertex b in the trust graph $G^* = (V^*, E^*)$ wants to construct a good approximation of G^* given the set N_b^* of neighbors of b in V^* and the set E_b^* of incoming edges of b in E^* .

The Setting The vertices in N_b^* and all the vertices under construction have $link_status \in \{linked, unlinked\}$ and $reply_status \in \{not_replied_yet, replying, replied\}$.

Initially $link_status(x) := linked$, and $reply_status(x) := not_replied_yet$, for all $x \in N_b^*$. Each (non-faulty) vertex $v \neq b$ in V^* makes an ordered list $edge_list(v)$ of its incoming edges. Each time v replies to a query of b requesting its incoming edges, v will send the first edge $first_edge(edge_list(v))$ in this list, and then remove this edge from the list. Edges are sent one-at-a-time. Initially $edge_list(v) := E_v^*$, where E_v^* is the complete list of incoming edges of v in E^* . Finally, a label of an edge (w,v) is $label_(w,v) := c_{wv}$, the certificate in which v certifies the public key of w.

```
Protocol (input = (b, N_b^*, E_b^*))
until is\_graph(G'') = satisfactory do
   begin
     b floods a query: "all vertices v send a label(w, v) of a new edge (w, v)";
     vertex v \neq b
        if |edge\_list(v)| > 1 then reply\_status(v) = replying;
        if |edge\_list(v)| = 1 then reply\_status(v) = replied;
        packet_{\bullet}flood to b: (data(v), sign_{k_{\bullet}}(data(v))),
                      where data(v) = (first\_edge(edge\_list(v)), reply\_status(v));
        edge\_list(v) := edge\_list(v) - first\_edge(edge\_list(v)) :
     vertex b
        if there is a path from b to v in G'' then link\_status(v) := linked
           else link\_status(v) := unlinked;
        if link\_status(v) := linked and (data(v), sign_{k_n}(data(v))) is correct
          and label(v, w) is correct then add\_to\_G''(label(v, w));
   end:
   G''' := missing\_edges(G'');
   G' := clean\_up(G''')
```

5 Proofs

We shall now prove that this protocol is efficient.

Lemma 2. The construction above will halt in $O(n^4)$ communication time, where n is the order of the trust graph G^* .

Proof. Let x be any vertex in G^* and let $\pi = (x = x_1, x_2, ..., x_r = b)$ be a path in G^* all of whose vertices are not faulty. The time taken to send the label of an edge (x,y) through the channel (x_i,x_{i+1}) of π is bounded by the number of neighbors of x_{i+1} in V^* , which is bounded by n. The length of π is bounded by n, so b will get the label of (x,y) in time n^2 . Since x cannot have more than x incoming edges, x will get the complete list of edges x in time x. It follows that x will get a first approximation of x in time x by this time, of course, x may also get up to x n fake vertices.

Next let us consider the cost of the halting and clean up tests. For the halting test we use a Max Fow algorithm with time complexity $O(|V''|^{1/2}|E''|)$. In our case $|V''| = O(n^4)$, $|E''| = O(n^4)$. The complexity for this test is then $O(n^2 \cdot n^4) \cdot O(n^4) = O(n^2 \cdot n^4 \cdot n^4) = O(n^{10})$. A similar argument applies for the clean up test with bound $O(n^{10})$. Observe that one should distinguish between the first complexity and the other two. The first one involves communication in the network, whereas the others are essentially off-line.

Lemma 3. The constructed graph G' is a good approximation of the trust graph G^* .

Proof. Suppose that a vertex x in G^* is not in the constructed graph G'. Let π_i , i = 1, 2, ..., 2u + 1, be vertex-disjoint directed paths in G^* from x to b, and let

 $x_i \in \pi_i$, i = 1, 2, ..., 2u+1, be vertices in G' whose ancestors in π_i are not in G'. The reply_status of the vertices x_i must be replying. Apply the halting test to these vertices to get a contradiction. The missing_edges routine guarantees that no edges are missing.

We therefore have,

Theorem 1. If the trust graph is $\lfloor 5u/2 + 1 \rfloor$ vertex-connected and if all the other assumptions in Section 2 hold, then the Protocol above will construct a good approximation of the trust graph in polynomial time.

6 Discussion

6.1 Reliability and the Communication Network

In our model we assume that the edges of the trust graph $G^* = (V^*, E^*)$ correspond to reliable channels. These can be regarded as virtual channels in a communication network $\bar{G} = (\bar{V}, \bar{E})$ for which $V^* \subset \bar{V}$. An edge $(v, w) \in E^*$ could correspond to some path in \bar{G} linking v, w, but not necessarily to a fixed path. Alternatively it could correspond to several paths, possibly vertex-disjoint. This would allow for the possibility of the channel (v, w) not being reliable, but reliability for the system may still be achieved through other paths in G^* .

This model is more general. However, at a high level we can add virtual edges whenever we get a reliable channel. So there is no real difference. Indeed we could argue that the general goal of secure communication is to extend the trust graph to a completely connected graph.

6.2 Identifying Labels of Processors

The following is an interesting scenario. A faulty processor may try to use different public keys, that is pseudonyms. The other faulty processors may be willing to support it, but a more successful strategy would be to get some non-faulty processors to accept the pseudonym. The effect of this would be to give the adversary more power, by controlling more parties. If no more than u non-faulty processors have certified a pseudonym p and if the trust graph is known then there is no problem. The pseudonym can be traced by observing that the connectivity $c(p,b) \leq 2u$ for every non-faulty processor b.

However if the trust graph is not known then there is a problem. This is because is not possible in our general adversarial model to construct a good approximation graph given only a vertex $b \in V^*$, the neighbor set N_b^* , and the edge set E_b^* . Indeed if the adversary controls (u+1) processors (u faulty processors and a pseudonym) then there is no way of preventing her from taking over the construction (in the general setting of Section 4).

This remark also suggests a method for extending the trust graph. A new processor will be "allowed in" if at least (2u + 1) processors are prepared to certify it. In the final version of the paper we will discuss this issue in more detail.

6.3 Double Certificates – Undirected Trust Graphs

In our model the edges (v, w) correspond to single certificates $c_{vw} = (v, w, k_v, k_w, sign_{k_w}(v, k_v))$ in which w certifies v's public key. If v also certifies w's public key with the certificate $c_{wv} = (w, v, k_w, k_v, sign_{k_v}(w, k_w))$ then we get an undirected edge (v, w). So the trust graph is undirected. All our results can be extend to this case.

References

- Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for noncryptographic fault-tolerant distributed computation. In Proceedings of the twentieth annual ACM Symp. Theory of Computing, STOC (May 2–4, 1988) pp. 1–10 275
- Bertsekas, D., Gallager, R.: Data networks second ed. Prentice Hall 1992 275, 279, 280
- Beth, T., Borcherding, M., Klein, B.: Valuation of trust in open networks. In Computer Security—ESORICS 94 (Lecture Notes in Computer Science 875) (1994) Springer-Verlag pp. 3–18 275
- Burmester, M., Desmedt, Y. G.: Secure communication in an unknown network with Byzantine faults. Electronics Letters 34 (1998) 741–742 275
- Burmester, M., Desmedt, Y., Kabatianskii, G.: Trust and security: A new look at the Byzantine generals problem. In Network Threats, DIMACS, Series in Discrete Mathematics and Theoretical Computer Science, December 2–4, 1996, vol. 38 (1998) R. N. Wright and P. G. Neumann, Eds., AMS 274, 275, 276, 277
- Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In Proceedings of the twentieth annual ACM Symp. Theory of Computing, STOC (May 2–4, 1988) pp. 11–19 275
- 7. Dolev, D.: The Byzantine generals strike again. Journal of Algorithms $\bf 3$ (1982) 14--30 $\, \, {\bf 275}$
- 8. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. Journal of the ACM 40 (1993) 17–47 275
- Even, S.: Graph algorithms. Computer science press Rockville, Maryland 1979 277, 281
- Franklin, M., Wright, R.: Secure communication in minimal connectivity models. In Advances in Cryptology — Eurocrypt '98, Proceedings (Lecture Notes in Computer Science 1403) (1998) K. Nyberg, Ed. Springer-Verlag pp. 346–360 275
- Franklin, M. K., Yung, M.: Secure hypergraphs: Privacy from partial broadcast. In Proceedings of the twenty seventh annual ACM Symp. Theory of Computing, STOC (1995) pp. 36–44 275
- Goldreich, O., Goldwasser, S., Linial, N.: Fault-tolerant computation in the full information model. SIAM J. Comput. 27 (1998) 506–544 275
- Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In Proceedings of the Nineteenth annual ACM Symp. Theory of Computing, STOC (May 25–27, 1987) pp. 218–229 275
- Hadzilacos, V.: Issues of Fault Tolerance in Concurrent Computations. PhD thesis Harvard University Cambridge, Massachusetts 1984 275
- Kaufman, C., Perlman, R., Speciner, M.: Network Security. Prentice-Hall Englewood Cliffs, New Jersey 1995 280

- Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Transactions on programming languages and systems 4 (1982) 382–401 275
- Maurer, U.: Modeling public-key infrastructure. In Computer Security— ESORICS 96 (Lecture Notes in Computer Science 1146) (1996) Springer-Verlag pp. 325–350 275
- Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. Journal of ACM 27 (1980) 228–234 275
- Popek, G. J., Kline, C. S.: Encryption and secure computer networks. ACM Computing Surveys 11 (1979) 335–356 275, 276
- Reiter, M. K., Stubblebine, S. G.: Path independence for authentication in large scale systems. In Proceedings of the 4th ACM Conference on Computer and Communications Security (April 1997) pp. 57–66 274, 275, 276
- 21. Rivest, R. L., Lampson, B.: SDSI-a simple distributed security infrastructure. http://theory.lcs.mit.edu/fcis/sdsi.html 276
- Wang, Y., Desmedt, Y.: Secure communication in broadcast channels. In Advances in Cryptology Eurocrypt '99, Proceedings (Lecture Notes in Computer Science 1592) (1999) J. Stern, Ed. Springer-Verlag pp. 446–458 275
- Zimmermann, P. R.: The Official PGP User's Guide. MIT Press Cambridge, Massachussets 1995 276

Linear Complexity versus Pseudorandomness: On Beth and Dai's Result

Yongge Wang*

Center for Applied Cryptographic Research, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada ygwang@cacr.math.uwaterloo.ca

Abstract. Beth and Dai studied in their Eurocrypt paper [1] the relationship between linear complexity (that is, the length of the shortest Linear Feedback Shift Register that generates the given strings) of strings and the Kolmogorov complexity of strings. Though their results are correct, some of their proofs are incorrect. In this note, we demonstrate with a counterexample the reason why their proofs are incorrect and we prove a stronger result. We conclude our note with some comments on the use of the LIL test (the law of the iterated logarithm) for pseudorandom bits generated by pseudorandom generators.

1 Introduction

Feedback shift registers, in particular linear feedback shift registers, are the basic components of many keystream generators. And the linear complexity of strings is an important tool to study different kinds of linear or nonlinear feedback registers.

Kolmogorov [6], and Chaitin [2] introduced the notion of Kolmogorov-Chaitin complexity of strings, which measures the minimum size of the input to a fixed universal Turing machine to generate the given string. While this complexity measure is of theoretical interest, there is no algorithm for computing it (it is equivalent to the problem of deciding whether a Turing machine halts on a given input, whence it is unsolvable).

Beth and Dai studied in their Eurocrypt paper [1] the relationship between linear complexity (that is, the length of the shortest Linear Feedback Shift Register that generates the given string) of strings and the Kolmogorov complexity of strings. Though their results are correct, some of their proofs are incorrect. In this note, we demonstrate the reason why their proofs are incorrect and we prove a stronger result.

Many stream ciphers utilize deterministically generated "random" sequences to encipher the message stream. Since the security of the system is based on the "randomness" of the key sequences, the criterion to determine the degree of "randomness" is crucial. In cryptographic community, the "randomness postulate"

^{*} Most of this work was done when the author was a post-docs at the University of Wisconsin-Wilwaukee.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 288–298, 1999. © Springer-Verlag Berlin Heidelberg 1999

by Golomb [5] and the "universal test" by Maurer [10] have gained widespread popularity. In Rueppel [13], [14, Ch.4], and Niederreiter [12], linear complexity profiles are proposed as a test for randomness. However, these tests cannot detect certain weaknesses in "pseudorandom sequences". For example, it should not be the case that the number of 1's in the initial segments of a random sequence is always larger (or equal) than the number of 0's. And it is easy to see that none of the above mentioned popular "tests" can detect this "weakness" in a sequence. At the end of this note, we suggest the use of the LIL test (the law of the iterated logarithm) as an additional test for pseudorandom sequences. The LIL test can detect several weaknesses in a pseudorandom sequence including the one mentioned above.

We close this section with some notation we will use. $\{0,1\}^*$, $\{0,1\}^n$, and $\{0,1\}^\infty$ are the set of finite binary strings, the set of binary strings of length n, and the set of infinite binary sequences respectively. The length of a string s is denoted by |s|. For a sequence $s \in \{0,1\}^* \cup \{0,1\}^\infty$ and an integer number $n \ge 1$, s[1..n] denotes the initial segment of length n of s (s[1..n] = s if $|s| \le n$) while s[n] denotes the nth bit of s, i.e., $s[1..n] = s[1] \dots s[n]$.

For a set $\mathbf{C} \subseteq \{0,1\}^{\infty}$ of infinite sequences, $Prob[\mathbf{C}]$ denotes the probability that $s \in \mathbf{C}$ when s is chosen by a random experiment in which an independent toss of a fair coin is used to decide the value of each bit in s. This probability is defined whenever \mathbf{C} is measurable under the usual product measure on $\{0,1\}^{\infty}$ (which can also be considered as the unit interval on the real number line).

2 Definitions and Basic Results

2.1 Linear Feedback Shift Registers

Linear feedback shift registers (LFSR) are the basic components of many keystream generators (see, e.g., Menezes et al. [11]). There are several reasons for this: LFSRs are well-suited to hardware implementation; they can produce sequences of large period; and they can produce sequences of good statistical properties.

A linear feedback shift register (LFSR) of length l is a sequence of 0-1 bits $(s_1, \ldots, s_l, c_1, \ldots, c_l)$ with $c_1 = 1$. The output of the LFSR is the infinite sequence $s_1 s_2 s_3 \ldots$ where s_i for i > l is defined by the following equation:

$$s_i = \sum_{j=1}^{l} c_j s_{i-l-1+j} \mod 2.$$

An LFSR $L(s_1, \ldots, s_l, c_1, \ldots, c_l)$ is said to generate an infinite sequence $s = s_1 s_2 \ldots$ if s is the output sequence of $L(s_1, \ldots, s_l, c_1, \ldots, c_l)$. The linear complexity of an infinite sequence s, denoted L(s), is defined as follows:

- 1. If s is the zero sequence 000..., then L(s) = 0.
- 2. If no LFSR generates s then $L(s) = \infty$.
- 3. Otherwise L(s) is the length of the shortest LFSR that generates s.

For a finite string $s \in \{0,1\}^n$, the linear complexity L(s) of s is defined as the length of the shortest LFSR that generates a sequence having s as its first n bits.

Theorem 1. (see, e.g., [9,11])

- 1. For $s \in \{0,1\}^n$, $0 \le L(s) \le n$.
- 2. For $s \in \{0,1\}^n$, L(s) = 0 if and only if $s = 0 \dots 0$.
- 3. For $s \in \{0,1\}^n$, L(s) = n if and only if $s = 0 \dots 01$.
- 4. If s is periodic with period n, then $L(s) \leq n$.

Theorem 2. For any strings $s_1, s_2, s_3 \in \{0, 1\}^*$, $L(s_1s_2s_3) \geq L(s_2)$.

Proof. This is straightforward from the definitions.

Theorem 3. (Massey [9]) For any given string $s \in \{0,1\}^n$, the Berlekamp-Massey algorithm will compute L(s) in $O(n^2)$ bit operations.

Theorem 4. For any $s \in \{0,1\}^n$, either $L(s0) \ge n/2$ or $L(s1) \ge n/2$.

Proof. In the proof of Theorem 3 (see [9,11]), it has been shown that if the shortest LFSR for generating s is $L(s_1, \ldots, s_l, c_1, \ldots, c_l)$, then there are two cases:

1. L(s0) = l. Then

$$L(s1) = \begin{cases} l & \text{if } l > n/2, \\ n+1-l & \text{otherwise.} \end{cases}$$

2. L(s1) = l. Then

$$L(s0) = \begin{cases} l & \text{if } l > n/2, \\ n+1-l & \text{otherwise.} \end{cases}$$

This completes the proof of the theorem.

Theorem 5. (Rueppel [14]) Let $k \le n$ and $N_n(k) = |\{s \in \{0,1\}^n : L(s) = k\}|$. Then

$$N_n(k) = \begin{cases} 2^{\min(2n-2k,2k-1)} & \text{if } n \ge k > 0, \\ 1 & \text{if } k = 0. \end{cases}$$

2.2 Kolmogorov Complexity

Kolmogorov complexity, as developed by Chaitin [2] and Kolmogorov [6] gives a satisfactory theoretical description of the complexity of individual finite strings and infinite sequences. In this section, we review the fundamentals of Kolmogorov complexity theory that we will use in this paper. For more details, it is referred to Li and Vitanyi [7]. Let U be a universal Turing machine. Then the Kolmogorov complexity of a string $s \in \{0,1\}^n$ is defined by

$$K(s) = \min\{|x|: U(x) = s, x \in \{0,1\}^*\}.$$

We are also interested in the self-delimiting Turing machines. A self-delimiting Turing machine is a deterministic Turing machine M such that the program set

$$PROG_M = \{x \in \{0,1\}^* : M \text{ halts on input } x \text{ after finitely many steps}\}$$

is prefix-free, i.e., a set of strings with the property that no string in it is a proper prefix of another. Let U_c be a universal self-delimiting Turing machine, then the Chaitin-Kolmogorov complexity of a string $s \in \{0,1\}^n$ is defined by

$$H(s) = \min\{|x| : U_c(x) = s, x \in \{0, 1\}^*\}.$$

Theorem 6. (see, e.g., [2,6,7])

- 1. There is a constant c > 0 such that $K(s) \leq H(s) + c$ for all $s \in \{0, 1\}^*$.
- 2. There is a constant c > 0 such that $H(s) \leq K(s) + 2\log|s| + c$ for all $s \in \{0, 1\}^*$.
- 3. For each Turing machine M and each string $s \in \{0,1\}^*$, let $K_M(s) =$ $\min\{|x|: M(x) = s, x \in \{0,1\}^*\}$. Then there is a constant $c_M > 0$ such that $K(s) \leq K_M(s) + c_M$ for all $s \in \{0, 1\}^*$.
- 4. There is a constant c > 0 such that $K(s) \le |s| + c$ for all $s \in \{0, 1\}^*$. 5. For any constant c > 0, $|\{s \in \{0, 1\}^n : K(s) \ge n c\}| \ge 2^{n c 1}$.

An infinite sequence $s \in \{0,1\}^{\infty}$ is Martin-Löf random (see, e.g., [7,8]) if and only if there is a constant c > 0 such that $H(s[1..n]) \ge n - c$ for almost all n.

Lemma 1. Let $s \in \{0,1\}^{\infty}$ be a Martin-Löf random sequence. Then there is a constant c > 0 such that $n - 2\log n - c \le K(s[1..n]) \le n + c$ for almost all n.

Proof. It follows from Theorem 6 and the definition of a Martin-Löf random sequence.

Theorem 7. (see, e.g., [7]) $Prob\{s: s \text{ is } Martin-L\"{o}f \text{ } random\} = 1.$

3 Linear Complexity versus Kolmogorov Complexity

Beth and Dai [1] proved the following theorem on the relationship between linear complexity and Kolmogorov complexity.

Theorem 8. (Beth and Dai [1]) For all $\varepsilon(0 < \varepsilon < 1)$

$$P_{\varepsilon,n} = Prob\{s \in \{0,1\}^n : (1-\varepsilon) \cdot 2L(s) \le K(s) \le (1+\varepsilon) \cdot 2L(s)\} \to 1$$

when $n \to \infty$.

After proving the above result, Beth and Dai [1] "proved" the following result:

Theorem 9. ([1]) With probability 1, a sequence $s \in \{0,1\}^{\infty}$ satisfies the following property:

$$\lim_{n \to \infty} \frac{K(s[1..n])}{L(s[1..n])} = 2. \tag{1}$$

Their proof is as follows:

"Proof". Apply the Borel-Cantelli lemma to the independent cylinder sets

$$\begin{split} A_{k,\varepsilon} &= \{s \in \{0,1\}^{\infty} : (1-\varepsilon) \cdot 2L(s[2^{k-1}..2^k-1]) \leq \\ &\quad K(s[2^{k-1}..2^k-1]) \leq (1+\varepsilon) \cdot 2L(s[2^{k-1}..2^k-1]) \} \end{split}$$

for the positive integers k and $\varepsilon > 0$. From Theorem 8, we conclude that

$$\sum_{k=1}^{\infty} Prob[A_{k,\varepsilon}] = \infty$$

for the positive integers k and $\varepsilon > 0$. Thus the assertion.

In the following we will show that the above "proof" is incorrect. Note that the second Borel-Cantelli lemma states as follows.

Theorem 10. (The second Borel-Cantelli Lemma [4]) Let $C_1, C_2, \ldots \subseteq \{0, 1\}^{\infty}$ be a sequence of independent, Lebesgue measurable sets, i.e.,

$$Prob[\mathbf{C}_i] \cdot Prob[\mathbf{C}_i] = Prob[\mathbf{C}_i \cap \mathbf{C}_i]$$

for $i \neq j$, such that $\sum_{k=1}^{\infty} Prob[\mathbf{C}_k]$ diverges. Then

$$\mathbf{C} = \{s \in \{0,1\}^{\infty} : s \in \mathbf{C}_k \text{ for only infinitely many } k\}$$

has probability 1.

By the Borel-Cantelli Lemma and the assertion $\sum_{k=1}^{\infty} Prob[A_{k,\varepsilon}] = \infty$, we can only infer in the above "proof" of Beth and Dai that

$$\mathbf{A} = \{ s \in \{0, 1\}^{\infty} : s \in A_{k, \varepsilon} \text{ for infinitely many } k \}$$

has probability 1. However, from this result we cannot infer that the equation (1) holds for all $s \in \mathbf{A}$, since there is an infinite sequence s such that $s \in A_{k,\varepsilon}$ for infinitely many k, but the equation (1) does not hold for s. Whence, Beth and Dai's "proof" is incorrect.

Indeed, we can construct an infinite sequence s such that $s \in A_{k,\varepsilon}$ for almost all k, but the equation (1) does not hold for s. In order to construct such a sequence s, we first prove two preliminary results.

Lemma 2. Let k and n be two positive integers. Then for any string $s \in \{0,1\}^{kn}$ such that L(s) = n, $K(s) \le 2n + 2\log k + c$ for some constant c > 0.

Proof. Define a Turing machine M_L as follows:

$$M_L(x) = \begin{cases} t[1..il] & \text{if } x = (i, s_1 \dots s_l c_1 \dots c_l) \text{ and the LFSR} \\ L(s_1, \dots, s_l, c_1, \dots, c_l) \text{ generates the infinite sequence } t, \\ \text{undefined otherwise.} \end{cases}$$

By Theorem 6, there is a constant c > 0 such that for all $s \in \{0,1\}^*$

$$K(s) \le \min\{|x| : M_L(x) = s\} + c.$$

For any string $s \in \{0,1\}^{kn}$ such that L(s) = n, let $L_s(s_1,\ldots,s_n,c_1,\ldots,c_n)$ be the shortest LFSR which generates a sequence having s as its first kn bits. By the definition of M_L , we have $M_L(k, s_1 \dots s_n c_1 \dots c_n) = s$. That is,

$$K(s) \le |(k, s_1 \dots s_n c_1 \dots c_n)| + c \le 2n + 2\log k + c.$$

This completes the proof.

Theorem 11. There is a constant $k_0 > 0$ such that for each $k > k_0$, there is a string $s_k \in \{0,1\}^{2^{k-1}}$ with the following properties:

- 1. $2L(s_k) c \le K(s_k) \le 2L(s_k) + c$ for some constant c > 0. 2. $L(s_k s_{k+1}[1]) \ge 2^{k-2}$.

Proof. We define the strings s_k by induction on k. Let k_0 be a large enough constant, $|s_{k_0}| = 2^{k_0-2}$ be a string with $L(s_{k_0}) = 2^{k_0-3}$, and $k > k_0$. Assume that s_{k-1} has already been defined. Then, by Theorems 4, 5, and 6, there is a string $s'_k \in \{0,1\}^{2^{k-3}}$ such that

- $\begin{array}{l} -2L(s_k')=2^{k-3};\\ -K(s_k')\geq 2^{k-3}-c_0 \text{ for some constant } c_0;\\ -L(s_{k-1}s_k'[1])\geq 2^{k-3}. \end{array}$

Let $s_k \in \{0,1\}^{2^{k-1}}$ be any string with the properties that $s_k' = s_k[1..2^{k-3}]$ and $L(s_k) = L(s'_k)$. By Lemma 2,

$$K(s_k) \le 2L(s_k) + 2\log 4 + c_1$$

for some constant $c_1 > 0$. It is straightforward to check that

$$K(s_k) \ge K(s'_k) - c_2 = 2^{k-3} - c_2 - c_0 = 2L(s_k) - c_2 - c_0$$

for some constant $c_2 > 0$. This completes the proof of the theorem.

Theorem 12. There is an infinite sequence s such that $s \in A_{k,\varepsilon}$ for almost all k, but the equation (1) does not hold for s.

Proof. Let k_0 and s_k be defined as in Theorem 11. Define an infinite binary sequence s as follows:

$$s[2^{k-1}..2^k - 1] = \begin{cases} 0...0 & \text{if } k \le k_0, \\ s_k & \text{otherwise.} \end{cases}$$

Then $s \in A_{k,\varepsilon}$ for all $k > k_0$. Now we show that the equation (1) does not hold for s. It is straightforward to check that for any $k > k_0$,

$$K(s[1..2^k]) \le \sum_{i=k_0+1}^k K(s_k) + c \le 2^{k-2} + c$$

for some constant c > 0. However, by the choice of s_k in Theorem 11, for any $k > k_0$,

$$L(s[1..2^k]) \ge L(s_k s_{k+1}[1]) \ge 2^{k-2}$$
.

Whence

$$\limsup_{n \to \infty} \frac{K(s[1..n])}{L(s[1..n])} \le \lim_{k \to \infty} \frac{2^{k-2} + c}{2^{k-2}} = 1 < 2.$$

Which implies that the equation (1) can not hold for s.

Theorem 12 shows that Beth and Dai's "proof" of Theorem 9 is incorrect. In the following we will prove a stronger result which implies Theorem 9. We first prove several lemmas.

Lemma 3. For any positive integer n,

$$|\{s \in \{0,1\}^n : L(s) \le \frac{n}{2} - \log n\}| \le 2^{n-2\log n}$$

and

$$|\{s \in \{0,1\}^n : L(s) \geq \frac{n}{2} + \log n\}| \leq 2^{n-2\log n}.$$

Proof. By Theorem 5, we have

$$|\{s \in \{0,1\}^n : L(s) \le \frac{n}{2} - \log n\}| = \sum_{k \le (n/2) - \log n} N_n(k)$$

$$= \sum_{k \le (n/2) - \log n} 2^{2k-1}$$

$$\le 2^{n-2\log n}$$

In the same way, we can show that $|\{s \in \{0,1\}^n : L(s) \ge \frac{n}{2} + \log n\}| \le 2^{n-2\log n}$.

Lemma 4. (see, e.g., [7]) Let $\mathbf{C}_1, \mathbf{C}_2, \ldots \subseteq \{0,1\}^{\infty}$ be a recursively presentable sequence of Lebesgue measurable sets, such that $\sum_{n=1}^{\infty} Prob[\mathbf{C}_n]$ converges to a finite number effectively. Then for any Martin-Löf random sequence $s \in \{0,1\}^{\infty}$, $s \in \mathbf{C}_n$ for only finitely many n.

Theorem 13. Let $s \in \{0,1\}^{\infty}$ be a Martin-Löf random sequence. Then the equation (1) holds for s.

Proof. Define a recursively presentable sequence of Lebesgue measurable sets as follows (for more details about recursively presentable sequence of sets, it is referred to Martin-Löf [8]): for each positive integer n, let

$$\mathbf{C}_n = \{ s \in \{0, 1\}^{\infty} : L(s[1..n]) \le \frac{n}{2} - \log n \text{ or } L(s[1..n]) \ge \frac{n}{2} + \log n \}.$$

Then, by Lemma 3,

$$Prob[\mathbf{C}_n] \le 2 \cdot 2^{-2\log n} = \frac{2}{n^2}.$$

Whence, $\sum_{n=1}^{\infty} Prob[\mathbf{C}_n]$ converges to a finite number effectively. By Theorem 4, for any Martin-Löf random sequence s, we have $s \in \mathbf{C}_n$ for only finitely many n. That is, $n/2 - \log n \le L(s[1..n]) \le n/2 + \log n$ for almost all n. Now, by Theorem 1,

$$\limsup_{n \to \infty} \frac{K(s[1..n])}{L(s[1..n])} \le \lim_{n \to \infty} \frac{n+c}{n/2 - \log n} = 2$$

and

$$\liminf_{n \to \infty} \frac{K(s[1..n])}{L(s[1..n])} \ge \lim_{n \to \infty} \frac{n - 2\log n}{n/2 + \log n} = 2.$$

This completes the proof of the theorem.

Proof of Theorem 9. This follows from Theorems 7 and 13.

Remark: Note that Niederreiter [12] has proved the following result: for any function f defined on the positive integers with the property that $\sum_{n=1}^{\infty} 2^{-f(n)} < \infty$, we have $Prob[\mathbf{C}_f] = 1$, where

$$\mathbf{C}_f = \left\{ s \in \{0, 1\}^{\infty} : \left| L(s[1..i]) - \frac{i}{2} \right| \le f(i) \ a.e. \right\}.$$

Whence, by Theorem 7 and the above result of Niederreiter, Theorem 9 follows. However, Theorem 13 is stronger than Theorem 9.

4 Comments on Statistical Tests

We conclude our note with some comments on statistical tests for pseudorandom bits generated by pseudorandom generators.

Martin-Löf randomness concept [8] has been the most successful one defined in the literature (see, e.g., [7]). By the Chaitin-Kolmogorov complexity characterization of Martin-Löf's random sequences, a sequence is Martin-Löf random if and only if it is incompressible, that is, if and only if the sequence withstands the compressibility test.

A Martin-Löf random sequence is defined to withstand all computational statistic tests (with unlimited resource bound). For example (see, e.g., [7]), a Martin-Löf random sequence withstands the frequency test (that is, the law of large numbers), the gap test, the correlation test, and the law of the iterated logarithm test. The result (that is, Theorem 13) of this paper can be interpreted in the following sense: a Martin-Löf random sequence withstands the LFSR tests. Amongst others, many of the statistic tests used for testing true randomness have been used to test the quality of the pseudorandom bits generated by a pseudorandom generator (see, e.g., [10,11]), for example, the frequency test, the gap test, the correlation test, and the Maurer universal test which is a kind of

compressibility test. However, a very strong test, the law of the iterated logarithm test (LIL test), has not been used in testing pseudorandom sequences. The celebrated iterated logarithm has been one of the most beautiful and profound discoveries (see, e.g., [3,4]) of probability theory. Wang [16,17,18] has shown that the law of the iterated logarithm holds for infinite polynomial time pseudorandom sequences. We suggest that this law should also be used in testing the quality of finite pseudorandom strings. By a standard diagonalization argument of Ville [15] for constructing "Kollektiv" sequences, it can be shown that there are sequences which pass the "randomness postulate" test (by Golomb [5]), the "universal test" (by Maurer [10]), and the LFSR test (by Rueppel [12,13,14]), but do not pass the LIL test. It should be mentioned that the LIL test can be finished in $O(n^2)$ time, whence it is an attractive addition to the currently used tests.

In the following we will present more details on the LIL test. For a sequence s, let $S_n(s) = \sum_{i=1}^n s[i]$. Then a sequence is said to withstand the frequency test if the value of $\frac{S_n(s)}{n}$ is close enough to $\frac{1}{2}$. However, this test is not successful in detecting whether a string s always has more 1's than 0's in its initial segments. Obviously, a pseudorandom sequence has some deficiency if there is always more 1's (or 0's) than 0's (or 1's) in its initial segments. As we have mentioned in the previous paragraph, Ville's construction can be used to show that all popular statistical tests used for pseudorandomness in the literature (see, e.g., [10,11]) cannot detect this deficiency which may have undesired effects in certain applications. We suggest the use of the law of the iterated logarithm (LIL). This test is "universal" in the sense that it covers many of the commonly used statistical tests such as the gap test and the frequency test, and in addition, it can detect the above mentioned deficiency in a sequence.

For a sequence s, let

$$S_n^*(s) = \frac{2 \cdot S_n(s) - n}{\sqrt{n}}$$

denote the reduced number of 1's in s[1..n]. Note that $S_n^*(s)$ amounts to measuring the deviations of $S_n(s)$ from $\frac{n}{2}$ in units of $\frac{1}{2}\sqrt{n}$. In probability theory, $S_n(s)$ is called the number of successes and $S_n^*(s)$ is called the reduced number of successes.

The law of large numbers says that, for a pseudorandom string s, the limit of $\frac{S_n(s[1..n])}{n}$ is $\frac{1}{2}$. But it says nothing about the reduced deviation $S_n^*(s[1..n])$. It is intuitively clear that, for a pseudorandom string s, $S_n^*(s[1..n])$ will sooner or later take on arbitrary large values. Moderate values of $S_n^*(s[1..n])$ are most probable, but the maxima will slowly increase. How fast? Can we give an optimal upper bound for the fluctuations of $S_n^*(s[1..n])$? The law of the iterated logarithm, which was first discovered by Khintchine for the classical cases, gives a satisfactory answer for the above questions.

Definition 1. A sequence $s \in \{0,1\}^{\infty}$ satisfies the law of the iterated logarithm if

$$\limsup_{n \to \infty} \frac{2\sum_{i=1}^{n} s[i] - n}{\sqrt{2n \ln \ln n}} = 1$$

and

$$\liminf_{n \to \infty} \frac{2\sum_{i=1}^{n} s[i] - n}{\sqrt{2n \ln \ln n}} = -1.$$

It has been shown that the law of the iterated logarithm holds for Martin-Löf random sequences (see [17]) and for infinite polynomial time pseudorandom sequences (see [16,17,18]). Since there is an efficient algorithm to compute the reduced number of successes in a string, a pseudorandom sequence $s \in \{0, 1\}^n$ of high "quality" should have the following LIL property: for large enough $i \le n$, the value of $\frac{2\sum_{j=1}^{i}s^{[j]-i}}{\sqrt{2i\ln\ln i}}$ should lie in the interval [-1-f(i),1+f(i)] for some function $f(i) \in o(\frac{1}{i})$ and the value should "reach" both 1 and -1 "frequently".

In the above paragraphs, we have given an outline of the LIL test for cryptographic pseudorandomness. In order to implement this test in practice, much work still needs to be done. For example, the distribution of the random variable $X_i = \frac{2\sum_{j=1}^{i} s[j]-i}{\sqrt{2i \ln \ln i}}$ should be carefully analyzed. For more details, it is referred to Chow [3] and Feller [4].

Acknowledgment

The author would like to thank one anonymous referee for his valuable report which helps improve the presentation of this paper.

References

- T. Beth and Z.-D. Dai. On the complexity of pseudo-random sequences or: If you can describe a sequence it cannot be random. In: Advances in Cryptology, Proc. of Eurocrypt '89, pp. 533-543, LNCS 434, Springer Verlag, 1989. 288, 291
- G. J. Chaitin. On the length of programs for computing finite binary sequences.
 J. Assoc. Comput. Mach., 13:547–569, 1966. 288, 290, 291
- 3. Y. S. Chow and H. Teicher. Probability Theory. Springer Verlag, 1997. 296, 297
- W. Feller. Introduction to Probability Theory and Its Applications. Volume I. John Wiley & Sons, Inc., 1968. 292, 296, 297
- S. W. Golomb. Shift Register Sequences. Holden-Day, San Francisco, Calif., 1967. 289, 296
- A. N. Kolmogorov. Three approaches to the definition of the concept "quantity of information". Problemy Inform. Transmission, 1:3-7, 1965. 288, 290, 291
- M. Li and P. Vitanyi. An Introduction to Kolmogorov Complexity and Its Applications. Springer, 1993. 290, 291, 294, 295
- P. Martin-Löf. The definition of random sequences. Information and Control, 9:602–619, 1966. 291, 294, 295
- J. L. Massey. Shift-register synthesis and BCH decoding. IEEE Transactions on Information Theory, 15:122–127, 1969.
- U. Maurer. A universal statistical test for random bit generators. Journal of Cryptology, 5:89–105, 1992. 289, 295, 296
- A. Menezes, P. van Oorschot, and S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996. 289, 290, 295, 296

- H. Niederreiter. The probability theory of linear complexity. In: Advances in Cryptology, Proc. of Eurocrypt '88, pp. 191–209, LNCS 330, Springer Verlag, 1989. 289, 295, 296
- R. Rueppel. Linear complexity and random sequences. In: Advances in Cryptology, Proc. of Eurocrypt '85, pp. 167–188, LNCS 219, Springer Verlag, 1986. 289, 296
- 14. R. Rueppel. Analysis and Design of Stream Ciphers. Springer, 1986. 289, 290, 296
- J. Ville. Ètude Critique de la Notion de Collectif. Gauthiers-Villars, Paris, 1939.
 296
- Y. Wang. The law of the iterated logarithm for p-random sequences. In: Proc. 11th Conference on Computational Complexity (formerly: Conference on Structure in Complexity Theory), pages 180-189. IEEE Computer Society Press, 1996. 296, 297
- Y. Wang. Randomness and Complexity. PhD thesis, Universität Heidelberg, 1996.
 296, 297
- Y. Wang. Resource bounded randomness and computational complexity. To appear in: Theoretical Computer Science, 1999. 296, 297

A Class of Explicit Perfect Multi-sequences

Chaoping Xing¹, Kwok Yan Lam², and Zhenghong Wei³

Department of Mathematics, National University of Singapore
 2 Science Drive 2, 117543, Singapore
 xingcp@math.nus.edu.sg
 School of Computing, National University of Singapore
 2 Science Drive 2, 117543, Singapore

lamky@comp.nus.edu.sg

Department of Mathematics, Shenzhen Normal College, People's Republic of China
weizhenghong@163.net

Abstract. In [7], perfect multi-sequences are introduced and a construction based on function fields over finite fields is given. In this paper, we explore the construction in [7] by considering rational function fields. Consequently a class of perfect multi-sequences are obtained.

1 Introduction

There are a lot of papers concerning linear complexity of one single sequence [2], [3], [4], [5], [8]. For multi-sequences, it seems that few results are known. One of these results is the Massey's algorithm for linear feedback shift register (LFSR) synthesis that was proven by Ding [1]. Recently perfect multi-sequences are introduced in [7]. This definition is a natural generalization of one single perfect sequence defined by Rueppel [4], [5].

We adopt notations from [7].

Definition 1. Let \mathbf{F}_q be the finite field with q elements. The linear complexity of a sequence $\mathbf{c} = (c_1, c_2, ..., c_n)$ of elements of \mathbf{F}_q is defined to be the least k such that it is a k-th order LFSR sequence.

We now consider multi-sequences $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_m$ of dimension m of length n. C denotes the multi-sequences $\{\mathbf{c}_i\}_{i=1}^m$ of length n.

Definition 2. The linear complexity of a multi-sequence set $C = \{\mathbf{c}_i = (c_{i1}, c_{i2}, \ldots, c_{in})\}_{i=1}^m$ of length n is defined to be the smallest order k of LFSR (f(T), k) generating all of sequences \mathbf{c}_i for $i = 1, 2, \ldots, m$.

The Massey's algorithm gives a way to find the shortest LFSR generating multi-seuqences, and therefore the linear complexity of a multi-sequence set C is obtained.

Now we turn to sequences of infinite length. Consider multi-sequences of dimension $m \geq 1$,

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 299–305, 1999.
 © Springer-Verlag Berlin Heidelberg 1999

$$\mathbf{a}_{1} = (a_{11}, a_{12}, a_{13}, \ldots) \in \mathbf{F}_{q}^{\infty}$$

$$\mathbf{a}_{2} = (a_{21}, a_{22}, a_{23}, \ldots) \in \mathbf{F}_{q}^{\infty}$$

$$\vdots$$

$$\mathbf{a}_{m} = (a_{m1}, a_{m2}, a_{m3}, \ldots) \in \mathbf{F}_{q}^{\infty}$$

and let A be the multi-sequence set $\{\mathbf{a}_i\}_{i=1}^m$. We also denote by A_n the multi-sequences

$$\{(a_{i1}, a_{i2}, \dots, a_{in})\}_{i=1}^m$$

of length n and dimension m.

Definition 3. The linear complexity profile of a multi-sequence set $A = \{\mathbf{a}_1, \mathbf{a}_2, \dots \mathbf{a}_m\}$ is defined by the sequence of integral numbers

$$\{\ell_n(A)\}_{n=1}^{\infty},$$

where $\ell_n(A)$ denote the linear complexities of $A_n = \{(a_{i1}, a_{i2}, \dots, a_{in})\}_{i=1}^m$ of length n.

In [7], perfect multi-sequences are defined in the following way.

Definition 4. The multi-sequence set $A = \{\mathbf{a}_1, \mathbf{a}_2, \dots \mathbf{a}_m\}$ is called perfect if

$$\ell_n(A) = \lceil \frac{mn}{m+1} \rceil$$

for all $n \ge 1$, where $\lceil v \rceil$ denotes the smallest integer bigger than or equal to the real number v.

Furthermore, it is proven in [7] that

Theorem 1. The multi-sequence set $A = \{\mathbf{a}_1, \mathbf{a}_2, \dots \mathbf{a}_m\}$ is perfect if and only if

$$\ell_n(A) \ge \lceil \frac{mn}{m+1} \rceil$$

for all $n \geq 1$.

This paper is organized as follows. In Section 2, we recall the main result of [7] concerning constructions of perfect multi-sequences. In Section 3, a class of perfect multi-sequences are given based on the result of Section 2. Some explicit examples are illustrated in this section.

2 Construction of Perfect Multi-sequences

First we introduce some notations (see [6]).

For a divisor $D = \sum_{P} m_{P} P$, the degree of D is defined by $\deg(D) = \sum_{P} m_{P} \deg(P)$.

Let z be a non-zero element of F, define the zero divisor of z by

$$(z)_0 = \sum_{\nu_P(z) > 0} \nu_P(z) P$$

and the *pole divisor* of z by

$$(z)_{\infty} = -\sum_{\nu_P(z)<0} \nu_P(z)P.$$

Then $(z)_0$ and $(z)_\infty$ are two effective divisors and $\deg(z)_0 = \deg(z)_\infty$. The principal divisor of z is defined by

$$\operatorname{div}(z) = (z)_0 - (z)_{\infty}.$$

For a divisor D, the linear space L(D) is defined by

$$L(D) = \{ f \in F | \operatorname{div}(f) + D \ge 0 \}.$$

It is an \mathbf{F}_q -linear space of finite dimension. For two divisors $D = \sum_P m_P P$ and $G = \sum_P n_P P$, we denote by $D \vee G$ the divisor $\sum_P \max\{m_P, n_P\}P$.

For a place Q of degree m of F, the integral ring of Q is

$$\mathcal{O}_Q = \{ z \in F | \nu_Q(z) \ge 0 \}.$$

This is a local ring with the maximal ideal

$$\wp_Q = \{ z \in \mathcal{O}_Q | \nu_Q(z) > 0 \}.$$

The residue class field \mathcal{O}_Q/\wp_Q is an extension over \mathbf{F}_q of degree $m = \deg(Q)$. For an element of $z \in \mathcal{O}_Q$, the residue class \overline{z} of z in \mathcal{O}_Q/\wp_Q is denoted by z(Q).

From now on in this section, we have the following notations and assumptions.

 F/\mathbf{F}_q – a global function field with the full constant field \mathbf{F}_q ;

Q – a place of degree m of F;

 $x_1, x_2, \ldots, x_m - m$ elements of \mathcal{O}_Q satisfying that $x_1(Q), x_2(Q), \ldots, x_m(Q)$ form an \mathbf{F}_q -basis of \mathcal{O}_Q/\wp_Q ;

t – a local parameter of Q with $deg(t)_{\infty} = m + 1$;

y – an element of \mathcal{O}_Q satisfying $y / \bigoplus_{i=1}^m \mathbf{F}_q(t) x_i$.

Then y has a local expansion at Q with the following form,

$$y = \sum_{j=0}^{\infty} \left(\sum_{i=1}^{m} a_{ij} x_i \right) t^j,$$

where $a_{i,j} \in \mathbf{F}_q$.

Denote

$$\mathbf{a}_i(y) = (a_{i1}, a_{12}, a_{i3}, \ldots) \in \mathbf{F}_q^{\infty}$$

for any $1 \le i \le m$. We form a multi-sequence set

$$A(y) = \{\mathbf{a}_i(y)\}_{i=1}^m. \tag{1}$$

It is proved in [7] that A(y) is a perfect multi-sequence set if $m = \deg((y)_{\infty} \vee (x_1)_{\infty} \vee (x_2)_{\infty} \vee \cdots \vee (x_m)_{\infty})$.

We organize the above result as follows.

Theorem 2. Let G be a positive divisor of degree m with Q /Supp(G). Suppose that $x_1, x_2, ..., x_m, y$ are elements of L(G) such that $x_1(Q), x_2(Q), ..., x_k(Q)$ form an \mathbf{F}_q -basis of \mathcal{O}_Q/\wp_Q and y / $\bigoplus_{i=1}^m \mathbf{F}_q(t)x_i$. Then $A(y) = \{\mathbf{a}_i(y)\}_{i=1}^m$ constructed as (1) is perfect.

3 Explicit Perfect Multi-sequences

In this section, we present a class of perfect multi-sequences based on the results of Section 2. The function fields used are the rational function fields.

Throughout this section, F denotes the rational function field $\mathbf{F}_q(x)$. We also fix a place Q of degree m and a rational place P of F with $Q \neq P$. Let $t \in F$ be a function with $(t)_0 = Q + P$.

Proposition 1. Let G be a positive divisor of degree m such that $Supp(G) \cap \{Q,P\} = \emptyset$. Let $\{x_1 = 1, x_2, ..., x_m, y\}$ be a basis of L(G). Then $\{x_1 = 1, x_2, ..., x_m, y\}$ is also a basis of F over $\mathbf{F}_q(t)$,

Proof. Since the degree $[F: \mathbf{F}_q(t)]$ of the extension $F/\mathbf{F}_q(t)$ is equal to the degree of the zero divisor of t, we have $[F: \mathbf{F}_q(t)] = \deg(t)_0 = \deg(Q+P) = m+1$. This means that F is a vector space of dimension m+1 over $\mathbf{F}_q(t)$. Hence it is sufficient to prove that $x_1 = 1, x_2, \ldots, x_m, y$ are linearly independent over $\mathbf{F}_q(t)$. Suppose that there exist m+1 functions $u_1(t), \ldots, u_m(t), u(t) \in \mathbf{F}_q(t)$ such that not all of them equal to zero and

$$u(t)y + \sum_{i=1}^{m} u_i(t)x_i = 0.$$
 (2)

By multipling a polynomial in $\mathbf{F}_q[t]$, we may assume that all $u_i(t)$ and u(t) are polynomials in $\mathbf{F}_q[t]$. Furthermore, dividing u(t) and all $u_i(t)$ by the greatest common divisor $(u(t), u_1(t), \ldots, u_m(t))$ of $u(t), u_1(t), \ldots, u_m(t)$, we can assume that $(u(t), u_1(t), \ldots, u_m(t)) = 1$. Thus not all of $u(0), u_1(0), \ldots, u_m(0)$ are equal to zero. As $x_1 = 1, x_2, \ldots, x_m, y$ are a basis of L(G), we have $u(0)y + \sum_{i=0}^m u_i(0)x_i \neq 0$.

Rewrite (2) into the form

$$(u(t) - u(0))y + \sum_{i=1}^{m} (u_i(t) - u_i(0))x_i = -\left(u(0)y + \sum_{i=1}^{m} u_i(0)x_i\right).$$
 (3)

Look at the left side of (3), we find that $Q + P = (t)_0$ is less than or equal to the zero divisor of the function $u(0)y + \sum_{i=1}^m u_i(0)x_i$. Hence $u(0)y + \sum_{i=1}^m u_i(0)x_i$ is an nonzero element of L(G - (Q + P)). However $L(G - (Q + P)) = \{0\}$ since $\deg(G - (Q + P)) = -1$. This contradiction shows that $x_1 = 1, x_2, \ldots, x_m, y$ are linearly independent over $\mathbf{F}_q(t)$.

Theorem 3. Let the divisor G and functions $x_1, x_2, ..., x_m, y$ be as in Proposition 1. If there is a positive divisor G_1 with $\deg(G_1) = m - 1$ such that $G_1 \leq G$ and $x_1, x_2, ..., x_m$ form a basis of $L(G_1)$, then the multi-sequence set A(y) constructed as (1) is perfect.

Proof. Since $\{x_1 = 1, x_2, \dots, x_m, y\}$ is a basis of F over $\mathbf{F}_q(t)$ by Proposition 1, we have $y / \bigoplus_{i=1}^m \mathbf{F}_q(t) x_i$.

Suppose that $\alpha_1, \alpha_2, ..., \alpha_m \in \mathbf{F}_q$ satisfy

$$\sum_{i=1}^{m} \alpha_i x_i(Q) = 0.$$

Then $\sum_{i=1}^{m} \alpha_i x_i$ is an element of $L(G_1 - Q)$. Thus $\sum_{i=1}^{m} \alpha_i x_i$ is equal to 0 since $\deg(G_1 - Q) = -1$. This implies that $x_1(Q), x_2(Q), ..., x_m(Q)$ form an \mathbf{F}_q -basis of \mathcal{O}_Q/\wp_Q .

Appling Theorem 2, we have that A(y) is perfect as deg(G) = m.

Example 1. Let Q(x) be an irreducible polynomials of degree m in $\mathbf{F}_q[x]$ and α an element of \mathbf{F}_q satisfying $Q(\alpha) \neq 0$. Put

- (a) $t = Q(x)(x \alpha)$;
- (b) $x_i = x^{i-1}$ for all $1 \le i \le m$;
- (c) $y = x^m$.

Then it follows from Theorem 3 that A(y) is a perfect multi-sequence set of dimension m since $\{x_1, x_2, ..., x_m\}$ is a basis of $L((m-1)\infty)$ and $\{x_1, x_2, ..., x_m, y\}$ is a basis of $L(m\infty)$, where ∞ is the pole of x.

(a) binary perfect multi-sequences of dimension m=3.

Taking $Q(x) = x^3 + x + 1 \in \mathbf{F}_2[x]$, t = xQ(x), and $x_1 = 1, x_2 = x, x_3 = x^2, y = x^3$, we obtain the local expansion of y at Q:

$$y = (x_1 + x_2) + (x_1 + x_3)t + (x_1 + x_2 + x_3)t^2 + x_1t^3 + (x_1 + x_2)t^4 + x_1t^5 + x_1t^6 + (x_1 + x_3)t^8 + \cdots$$

Put

$$\mathbf{a}_1(y) = (1, 1, 1, 1, 1, 1, 0, 1, \cdots),$$

$$\mathbf{a}_2(y) = (0, 1, 0, 1, 0, 0, 0, 0, \cdots),$$

$$\mathbf{a}_3(y) = (1, 1, 0, 0, 0, 0, 0, 1, \cdots).$$

By Theorem 3, $\mathbf{a}_1(y)$, $\mathbf{a}_2(y)$, $\mathbf{a}_3(y)$ are perfect multi-sequences of dimension 3.

(b) ternary perfect multi-sequences of dimension m=2.

Taking $Q(x) = x^2 + 1$, $t = x(x^2 + 1)$, and $x_1 = 1, x_2 = 2, y = x^2$, we get the local expansion of y at Q:

$$y = 2x_1 + 2x_2t + 2x_1t^2 + x_2t^3 + 2x_1t^4 + 2x_1t^6 + 2x_2t^9 + \cdots$$

Put

$$\mathbf{a}_1(y) = (0, 2, 0, 2, 0, 2, 0, 0, \cdots),$$

 $\mathbf{a}_2(y) = (2, 0, 1, 0, 0, 0, 0, 0, \cdots).$

By Theorem 3, $\mathbf{a}_1(y)$, $\mathbf{a}_2(y)$ are perfect multi-sequences of dimension 2.

Example 2. Let Q(x) be an irreducible polynomials of degree m in $\mathbf{F}_q[x]$ and α an element of \mathbf{F}_q satisfying $Q(\alpha) \neq 0$. Put

- (a) $t = Q(x)/x^{m+1}$;
- (b) $x_i = (x \alpha)^{1-i}$ for all $1 \le i \le m$;
- (c) $y = (x \alpha)^{-m}$.

Then it follows from Theorem 3 that A(y) is a perfect multi-sequence set of dimension m since $\{x_1, x_2, ..., x_m\}$ is a basis of L((m-1)P) and $\{x_1, x_2, ..., x_m, y\}$ is a basis of L(mP), where P is the zero of $x - \alpha$.

(a) binary perfect multi-sequences of dimension m=2.

Taking $Q(x) = x^2 + x + 1$, $t = (x^2 + x + 1)/x^3$, and $x_1 = 1$, $x_2 = 1/x$, $y = 1/x^2$, we obtain the local expansion of y at Q:

$$y = (x_1 + x_2) + (x_1 + x_2)t + x_2t^2 + (x_1 + x_2)t^3 + t^4 + t^6 + \cdots$$

Put

$$\mathbf{a}_1(y) = (1, 0, 1, 1, 0, 1, \cdots),$$

 $\mathbf{a}_2(y) = (1, 1, 1, 0, 0, 0, \cdots).$

By Theorem 3, $\mathbf{a}_1(y)$, $\mathbf{a}_2(y)$ are perfect multi-sequences of dimension 2.

(b) ternary perfect multi-sequences of dimension m=2.

Taking $Q(x) = x^2 + 2x + 2$, $t = (x^2 + 2x + 2)/x^3$, and $x_1 = 1, x_2 = 1/x, y = 1/x^2$, we obtain the local expansion of y at Q:

$$y = (x_1 + 2x_2) + (2x_1 + 2x_2)t + (x_1 + 2x_2)t^2 + x_1t^3 + (x_1 + 2x_2)t^4 + \cdots$$

Put

$$\mathbf{a}_1(y) = (2, 1, 1, 1, \cdots),$$

 $\mathbf{a}_2(y) = (2, 2, 0, 2, \cdots).$

By Theorem 3, $\mathbf{a}_1(y)$, $\mathbf{a}_2(y)$ are perfect multi-sequences of dimension 2.

References

- C. S. Ding, "Proof of Massey's conjectured algorithm," in Advances in Cryptology
 Eurocrypt'88, C. G. Guenther ed., Springer-Verlag, Berlin, LNCS Vol. 310, 1988,
 pp.345-349.
- D. R. Kohel, S. Ling and C. P. Xing, "Explicit sequence expansions," to appear in DMTC, Springer-Verlag. 299
- 3. H. Niederreiter, "Sequences with almsot perfect linear complexity profile," in *Advances in Cryptology–EUROCRYPT'87*, D. Chaumand and W. L. Price, eds., Springer-Verlag, Berlin: LNCS, Vol. 304, 1988, pp. 37-51. 299
- R. A. Rueppel, Analysis and Design of Stream Ciphers, Berlin: Springer-Verlag, 1986.
- R. A. Rueppel, "Stream ciphers," in Contemporary Cryptology-The Science of Information Integrity, G. J. Simmons, Ed., IEEE Press, New York: 1992, pp. 65-134.
- H. Stichtenoth, Algebraic Function Fields and Codes, Berlin: Springer-Verlag, 1993.
 301
- C. P. Xing, "Perfect multi-sequences and function fields over finite fields", Preprint, 1999. 299, 300, 302
- C. P. Xing and K. Y. Lam, "Sequences with almost perfect linear complexity profiles and curves over finite fields," *IEEE Trans. Inform. Theory*, vol. IT-45, 1999, pp.1267-1270. 299

Cryptanalysis of LFSR-Encrypted Codes with Unknown Combining Function

Sarbani Palit¹ and Bimal K. Roy²

Computer Vision & Pattern Recognition Unit, Indian Statistical Institute, 203, B T Road, Calcutta 700 035, INDIA sarbanip@isical.ac.in
Applied Statistics Unit, Indian Statistical Institute, 203, B T Road, Calcutta 700 035, INDIA bimal@isical.ac.in

Abstract. This paper proposes an approach for the cryptanalysis of stream ciphers where the encryption is performed by multiple linear feedback shift registers (LFSR) combined by a nonlinear function. The attack assumes no knowledge of either the LFSR initial conditions or the combining function. Thus, the actual architecture of the encryption system can be arbitrary. The attack is also generalized for the situation when the combining function is correlation immune of any particular order. This is in direct contrast with the existing methods which depend heavily not only on the correlation between the output of a particular LFSR and the ciphertext but also on the actual configuration of the encryption system used. Thus, the proposed method is the first *ciphertext only* attack in the true sense of the phrase. The paper also gives theoretical estimates of the cipherlengths involved in the determination of the initial conditions as well as estimation of the combining function.

1 Introduction

Stream ciphers form an important class of encryption algorithms. Linear feedback shift registers (LFSR) have commonly been used in the keystream generators of such ciphers for a reasons such as suitability for hardware implementation and desirable statistical properties [1]. A popular form of the running key generator is constructed by applying a nonlinear combining function to the outputs of several LFSRs of various lengths. In a general situation, the decrypter may be faced with one or more of the following problems, *viz.* unknown initial conditions of the LFSRs, unknown shift register lengths and polynomials, unknown combining function, availability of limited cipher length, and need for computation in reasonable time,

The problem of determining the unknown LFSR sequences or their initial conditions from available ciphertext using a correlation attack was first explored by Siegenthaler in [2]. This approach exploited the inherent weakness of keystreams generated using LFSRs where knowledge about the LFSR creeps into the encrypted data. It was followed by other forms of correlation attacks

as well as modifications for improvement in speed, see [3,4,5,6,7]. All these approaches, however, implicitly assume that the ciphertext is directly correlated to the LFSR sequences, *i.e.* the combining function is not correlation immune [8] and also known to the decrypter.

This paper presents a decryption strategy based on a modification of Siegenthaler's method. The method given here is developed in steps in sections 2 to 5 to include the cases of unknown LFSR initial conditions and unknown combining function, which is eventually allowed to be correlation immune of a particular order. The method is therefore, independent of the actual architecture of the encryption system, unlike the other existing methods in the literature. Though knowledge of the LFSR polynomials is assumed, this assumption can be dispensed with as explained later. The paper also presents a framework for determining the cipherlength requirements involved.

2 Determination of the Initial Conditions

Let N denote the cipherlength available, C the ciphertext, X_i the sequence produced by the ith LFSR, m the total number of LFSRs and d_i , the size of the ith LFSR. Also, P(A) denotes the probability of the occurrence of the event A. Let M be the coded message text, Y be the output of the combining function. We assume initially that the shift register sizes, the LFSR polynomials and the form of the nonlinear combining function are known.

2.1 Siegenthaler's Approach

Siegenthaler's approach [2] for determining the initial conditions of the LFSRs is based on statistical hypothesis testing. Consider the random sequence

$$Z_i = \begin{cases} 1 & \text{if } C = X_i \\ 0 & \text{if } C \neq X_i \end{cases}$$

It can thus be inferred that $\sum Z_i \sim Bin(N, p)$ where $p = P(C = X_i)$. From the system configuration

$$P(C = X_i) = P(Y = X_i)P(M = 0) + P(Y \neq X_i)P(M \neq 0)$$

If either $P(Y=X_i)=1/2$ or P(M=0)=1/2, then $P(C=X_i)=1/2$. However, for all practical coding schemes, $P(M=0)\neq 1/2$. For instance, for the popularly used Murray code P(M=0)=0.58. Also, for a function which is not correlation immune, there is at least one input i for which $P(Y=X_i)\neq 1/2$. In order to break the code, the LFSRs are run with various initial conditions. When the correct initial condition is used to generate the LFSR sequence X_i , $p=P(C=X_i)\neq 1/2$. In contrast, for a wrong initial condition, the sequence X_i is random and uncorrelated with C which implies that p=1/2. Thus, given

an input sequence X_i and C, we can determine whether the corresponding initial condition is correct by testing the following hypotheses:

$$\mathcal{H}_0: p = 0.5,$$

 $\mathcal{H}_1: p \neq 0.5.$

The statistic used for testing \mathcal{H}_0 against the alternative \mathcal{H}_1 is $\sum Z_i$. Let us assume that the fraction of coincidences between the cipher stream and an input i.e. $\sum_{N=1}^{N} Z_i$ is normally distributed. This assumption is justified by the Central Limit Theorem. According to the usual Neyman-Pearson set-up for hypothesis testing, the decision threshold for $\sum Z_i$ is chosen so that the probability of wrongly rejecting \mathcal{H}_0 is restricted to a specified value. However, Siegenthaler proposes that the threshold be chosen so that the probability of wrongly accepting \mathcal{H}_0 (probability of 'miss', p_m or 1-'power' in the terminology of hypothesis testing) is restricted to a specified value. The probability of the other type of error (probability of 'false alarm', p_f) is not controlled. It would depend on the nature of the combining function and the cipherlength.

2.2 Notion of Success and Failure

The above approach can fail in two ways: (a) the correct initial condition may be missed, or (b) there may be too many false alarms. The twin objective of the decision-making process is to restrict the chances of both types of failures. However, a precise definition of success is necessary in order to examine the feasibility of breaking a code with a given cipherlength. A reasonable approach would be to define 'success' as the situation when the correct initial condition belongs to the selected list of solutions along with k wrong initial conditions. The number k has to be chosen before any analysis of performance. A high value of k would mean that a large number of candidate solutions have to be examined by actually generating the 'deciphered' texts – a prospect which can hardly be described as 'success'. Thus, k has to be a reasonably small number. In this paper, we choose k = 0. The same choice was implicitly made by Roy [9]. Of course, all the calculations can be generalized for k > 0. Our choice of k implies that 'success' is defined as the case when the shortlist of selected initial conditions contains only the correct one.

2.3 Choice of Optimal Threshold to Maximize Chance of Success

Siegenthaler [2] suggested that the probability of miss (p_m) should be predetermined. In practice however, the choice of p_m must be a compromise to ensure that not too many wrong candidate solutions are selected. According to the notion of success described above, no false alarm is acceptable. Thus, instead of using a predetermined p_m , we can determine the threshold in such a way that both p_m and the probability of any false alarm is minimized. If the *i*th LFSR has size d_i , the probability of no false alarm is $(1-p_f)^{2^{d_i}-2}$, where p_f is the probability of

false alarm for a particular candidate solution. Note that if the decision threshold is moved in such a way that $1-(1-p_f)^{2^{d_i}-2}$ (the probability of having at least one wrong initial condition) is reduced, p_m would increase as a consequence. Therefore, the 'minimax' strategy of minimizing $\max\{p_m,\ 1-(1-p_f)^{2^{d_i}-2}\}$ leads to the solution $1-(1-p_f)^{2^{d_i}-2}=p_m$.

Suppose that δ_i is the 'separation' $P(C = X_i) - 0.5$. If $\delta_i > 0$, the decision rule is to classify an initial condition as 'correct' when $\frac{1}{N} \sum_{i=1}^{N} Z_i > t$, where t is the decision threshold. The minimax solution described above is equivalent to the following equation for t:

$$1 - \Phi\left(\frac{t - 0.5}{\sqrt{0.25/N}}\right)^{2^{d_i} - 2} = \Phi\left(\frac{t - (0.5 + \delta_i)}{\sqrt{((0.5 + \delta_i)(0.5 - \delta_i))/N}}\right),$$

where $\Phi(\cdot)$ is the standard normal distribution function.

A suitable modification of the above condition can be made when $\delta_i < 0$.

2.4 A Modification of Siegenthaler's Method

Let us assume once again that $\delta_i > 0$. Let f_{ic} be the observed fraction of coincidences, $\sum Z_i/N$, when the chosen initial condition is correct. Suppose that f_{iw} be the largest value of $\sum Z_i/N$ when a wrong initial condition is used. Siegenthaler's method would be successful (in the sense described in section 2.2) if $f_{iw} \leq t < f_{ic}$. However, f_{ic} can be larger than f_{iw} even if both of these are on the same side of t. A correct determination of the initial condition is possible in such a case, by modifying Siegenthaler's approach as follows. Let f_i be the fraction of coincidences between the ciphertext and the ith input. One may check for the maximum of f_i over all possible initial conditions. In the modified approach, the maximizer is identified as the correct initial condition of the ith input. If $\delta_i < 0$, the minimizer of f_i over all possible initial conditions should be identified as the correct initial conditions.

Henceforth we will refer to Siegenthaler's method with threshold chosen as in Section 2.3 as Method 1, and the 'best-is-correct' approach described in this section as Method 2.

2.5 Performance of the two Methods

Consider Method 1 with the threshold set at t, and assume $\delta_i > 0$ without loss of generality.

P(ith input is correctly determined)= $P(\text{the 'correct' LFSR sequences has } f_i > t$ AND all 'wrong' LFSR sequences have $f_i \leq t$),

where a 'correct' LFSR sequence corresponds to that generated using the correct initial condition (i.c.). Therefore,

 $P(ith input is correctly determined) = (1 - p_m)(1 - p_f)^{n_i - 1}$, where

$$\begin{aligned} 1-p_m &= P(\text{a 'correct' LFSR sequence has } f_i > t) \approx \varPhi\left(\frac{.5+\delta_i-t}{\sqrt{0.25/N}}\right), \\ 1-p_f &= P(\text{a 'wrong' LFSR sequence has } f_i \leq t) \approx \varPhi\left(\frac{t-.5}{\sqrt{0.25/N}}\right), \\ n_i &= \text{total no. of LFSR sequences for input } i = 2^{d_i} - 1. \end{aligned}$$

Note that the probabilities $(1 - p_m)$ and $(1 - p_f)^{n_i - 1}$ are the same because of the minimax strategy of section 2.3. Denoting m as the number of inputs,

$$P(\text{all inputs are correctly determined}) = \prod_{i=1}^{m} \Phi\left(\frac{.5 + \delta_i - t}{\sqrt{0.25/N}}\right)^2, \quad (1)$$

Next consider Method 2. Let

$$p_i = P(C = X_i | i \text{th LFSR i.c. is chosen correctly}).$$

, which can be computed for a particular combining function. The number of coincidences N_{ic} when the i.c. is chosen correctly has a binomial distribution: $N_{ic} \sim Bin(N, p_i)$. When the i.c. is chosen wrongly, the number of coincidences, $N_i \sim Bin(N, 0.5)$. There are $2^{d_i} - 2$ wrong initial conditions. Assuming that $\delta_i > 0$, we have to consider the probability that the maximum of the N_i 's over all these wrong i.c.'s, denoted here by N_{iw} , is not very large.

$$P((N_{iw} < y) = \left(\sum_{k=0}^{y} {N \choose k} p^k (1-p)^{N-k}\right)^{(2^{d_i}-2)} \approx \Phi\left(\frac{2y-N}{\sqrt{N}}\right)^{(2^{d_i}-2)}$$

for y = 0, 1, ..., N. It follows that,

$$P(i\text{th input is correctly identified}) = P((N_{iw} < N_{ic}))$$

$$= \sum_{y=0}^{N} \binom{N}{y} P((N_{iw} < y) p_i^y (1 - p_i)^{N-y})$$

$$= \sum_{y=0}^{N} \binom{N}{y} p_i^y (1 - p_i)^{N-y} \left\{ \Phi\left(\frac{2y - N}{\sqrt{N}}\right) \right\}^{(2^{d_i} - 2)}$$

It can be easily seen that the above calculations go through when $\delta_i < 0$.

The probabilities of correct identification of all the input i.c.s have to be multiplied in order to obtain the overall probability of correct identification. Figure 1 shows plots of the variation of cipherlength with the desired probability of correct identification of the initial condition of an LFSR having length 12 and 16, using both the methods for $\delta=0.02$. Figure 2 shows the corresponding plots for $\delta=0.06$. Note that,

- The cipherlength requirement to achieve a desired probability of correct identification, increases with increase in the size of the LFSR.
- As δ (the separation from 0.5) increases, the cipherlength requirement for a fixed probability of correct identification, reduces remarkably.

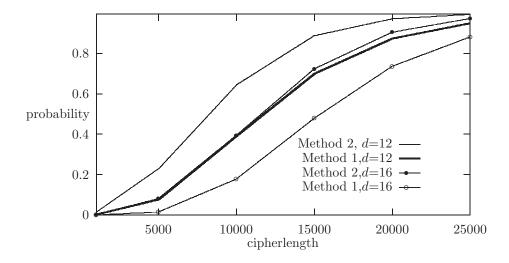


Figure 1: Probability of correct identification of an i.c. vs. cipherlength required, $\delta = 0.02$

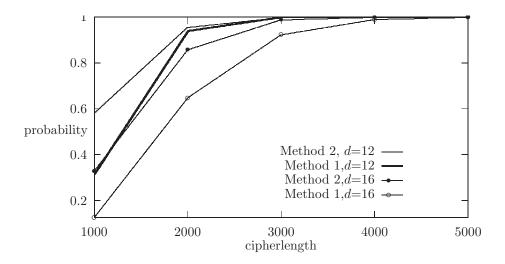


Figure 2: Probability of correct identification of an i.c. vs. cipherlength required, $\delta=0.06$

It can also be seen that the cipherlength requirements for Method 2 are much less than that of Method 1. For the rest of the paper, only Method 2, *i.e.* the 'best-is-correct' approach shall be adopted.

3 Estimation of the Combining Function

We now consider the case where the combining function is unknown, but the LFSR i.c.'s have been correctly identified. This would help us eventually in addressing the problem of identifying the initial conditions and the combining function when both are unknown. The latter case is considered in Section 4.

3.1 A Maximum Likelihood Approach

Identifying the combining function amounts to determining the 2^m binary output values in the corresponding truth table. We treat these numbers, denoted here by $Y_0, Y_1, \ldots, Y_{2^m-1}$, as unknown parameters. These parameters control the distribution of the cipher stream. Using the knowledge of the inputs and the cipherstream, we proceed to obtain the maximum likelihood estimate of these parameters.

Suppose that p_0 is the probability that a bit of the plain text stream is equal to 0. As mentioned earlier, for all practical coding schemes, $p_0 \neq 0.5$. Throughout this paper, we have used the Murray code, for which $p_0 = 0.58$. Given that the corresponding function output is Y_j , the *i*th bit C_i of the cipher stream has the following probability mass function:

$$C_i = \begin{cases} 1 & \text{with probability } 1 - p_0 + (2p_0 - 1)Y_j, \\ 0 & \text{with probability } p_0 - (2p_0 - 1)Y_j, \end{cases}$$

This can be written in a more compact form as

$$P(C_i = c|Y_i) = [1 - p_0 + (2p_0 - 1)Y_i]^c \cdot [p_0 - (2p_0 - 1)Y_i]^{1-c}, \quad c = 0, 1.$$

Let $I_0, I_1, \ldots, I_{2^m-1}$ be the sets of indices of the bitstream that correspond to the 2^m different input combinations, respectively. [Note that these input combinations correspond to the outputs $Y_0, Y_1, \ldots, Y_{2^m-1}$, respectively.] The sizes of these sets, $N_0, N_1, \ldots, N_{2^m-1}$, have a multinomial probability distribution with equal probabilities for each of the 2^m cells. The joint distribution of the cipherstream given the input streams is

$$\prod_{j=0}^{2^m-1} \prod_{i \in I_j} P(C_i = c_i | Y_j) = \prod_{j=0}^{2^m-1} \prod_{i \in I_j} [1 - p_0 + (2p_0 - 1)Y_j]^{c_i} \cdot [p_0 - (2p_0 - 1)Y_j]^{1 - c_i}.$$

Thus, the likelihood of $Y_0, Y_1, \ldots, Y_{2^m-1}$ given the input streams and the cipher-stream is

$$\ell(Y_0, Y_1, \dots, Y_{2^m - 1}) = \prod_{j = 0}^{2^m - 1} \prod_{i \in I_j} [1 - p_0 + (2p_0 - 1)Y_j]^{C_i} \cdot [p_0 - (2p_0 - 1)Y_j]^{1 - C_i}.$$

It may be noted that the parts that depend on each Y_j appear as factors of the overall likelihood. Thus, we can work with one 'likelihood function' for each Y_j ,

 $j = 0, 1, \dots, 2^m - 1$:

$$\ell_j(Y_j) = \prod_{i \in I_j} [1 - p_0 + (2p_0 - 1)Y_j]^{C_i} \cdot [p_0 - (2p_0 - 1)Y_j]^{1 - C_i}.$$

Therefore, the MLE of Y_j is

$$\widehat{Y}_j = \begin{cases} 0 \text{ if } \ell_j(0)/\ell_j(1) > 1\\ 1 \text{ otherwise} \end{cases}$$

The condition $\ell_j(0)/\ell_j(1) > 1$ reduces to $[(1-p_0)/p_0]^{\sum_{i \in I_j} (2C_i-1)} > 1$. When $p_0 > .5$, as is the case for the Murray code, this further simplifies to $\sum_{i \in I_j} C_i < N_i/2$.

In summary, the MLE of Y_j is

$$\widehat{Y}_j = \begin{cases} 0 & \text{if } \sum_{i \in I_j} C_i < N_j/2, \\ 1 & \text{if } \sum_{i \in I_j} C_i > N_j/2, \end{cases}$$
 $j = 0, 1, \dots, 2^m - 1.$

In the unlikely event when $\sum_{i \in I_j} C_i = N_j/2$, both 0 and 1 are MLE, and one can assign one or the other without loss of generality.

Thus, the algorithm for function estimation can be summarized in the following steps:

- 1. Select a particular input combination
- 2. Count the number of times Y = 1 for a particular input combination. Count the number of times this particular input combination occurs.
- 3. Compute the proportion of 1's from the above counts.
- 4. Conclude that output (Y) for that input combination is 0 if proportion of 1's is less than 0.5, and 1 otherwise.
- 5. Repeat steps 1-4 for another input combination until all combinations are exhausted.

3.2 Cipher Length Requirements

Suppose that for the jth input combination, the function output $Y_j=0$. Assume that the index set for this input combination is I_j (with N_j elements). Then Y_j is correctly identified if at most $N_j/2$ out of the N_j cipher bits corresponding to the index set I_j turn out to be 1. This in turn occurs when at most $N_j/2$ of the corresponding N_j bits of the plain text happen to be 1. The latter event has probability $\sum_{k=N_j/2}^{N_j} \binom{N_j}{k} p_0^k (1-p_0)^{N_j-k}$. It is easy to see that the above expression for the probability of correct identification of Y_j holds even when $Y_j=1$.

Using the fact that the numbers $N_0, N_1, \ldots, N_{2^m-1}$ have a multinomial distribution, we have

P(The entire m-input truth table is correctly identified)

$$= \sum_{N_0, N_1, \dots, N_{2^m - 1}} \frac{N!}{N_0! N_1! \dots N_{2^m - 1}!} \left(\frac{1}{2^m}\right)^N$$

$$\prod_{j=0}^{2^m - 1} \left(\sum_{k=N_j/2}^{N_j} \binom{N_j}{k} p_0^k (1 - p_0)^{N_j - k}\right)$$

The actual computation of this quantity can be extremely time-consuming, because of the multiple summations. Alternatively, one can ignore the dependence of the cell frequencies, and use a binomial distribution for each N_j , with probability of inclusion $1/2^m$. The (ignored) correlation of two cell frequencies is 2^{-2m} , which is small for $m \geq 3$. This leads to the following approximation:

P(The entire m-input truth table is correctly identified)

$$= \left\{ \sum_{k=0}^{N} \left(\binom{N}{k} \left(\frac{1}{2^m} \right)^k \left(1 - \frac{1}{2^m} \right)^{N-k} \left(\sum_{l=[k/2]}^k \binom{k}{l} p^l (1-p)^{k-l} \right) \right) \right\}^{2^m} (2)$$

Figure 3 shows the cipher length requirements vs. the probability of correct estimation of the function for a three-input and a five-input function. As expected, the cipher length required to achieve a particular probability of estimation is much more in the case of a five-input function than a three-input one.

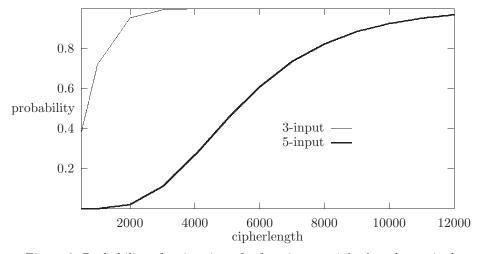


Figure 3: Probability of estimation of a function vs. cipherlength required

4 Determination of the Initial Conditions when the Combining Function Is Unknown

Let us assume that the combining function is not correlation immune with respect to any of its inputs. If the function and the i.c.s are unknown, the quantity $f_i - 0.5$ though still non-zero for the correct initial condition, may be either positive or negative. Hence, the 'best-is-correct' approach is slightly modified as follows. Check for the maximum of $|f_i - 0.5|$ over all possible initial conditions. Identify the maximiser as the correct i.c. of the *i*th input. Repeat for the other inputs. Once all the i.c.s are identified, estimate the function as outlined in the previous section.

Correspondingly, we can compute P(ith initial condition is correctly identified)

$$\begin{split} &= \sum_{y=0}^{N/2} \binom{N}{y} p_i^y (1-p_i)^{N-y} \left(\sum_{k=y}^{N-y} \binom{N}{k} (0.5)^k (0.5)^{N-k} \right)^{(2^{d_i}-2)} \\ &\quad + \sum_{y=N/2+1}^{N} \binom{N}{y} p_i^y (1-p_i)^{N-y} \left(\sum_{k=N-y}^{y} \binom{N}{k} (0.5)^k (0.5)^{N-k} \right)^{(2^{d_i}-2)} \\ &\approx \sum_{y=0}^{N} \binom{N}{y} p_i^y (1-p_i)^{N-y} \left[2 \varPhi \left(\frac{|N-2y|}{\sqrt{N}} \right) - 1 \right]^{2^{d_i}-2} \end{split}$$

So, P(all i.c.s are correctly identified),

$$P_{ic} = \prod_{i=1}^{m} \left[\sum_{y=0}^{N} {N \choose y} p_i^y (1 - p_i)^{N-y} \left\{ 2\Phi \left(\frac{|N - 2y|}{\sqrt{N}} \right) - 1 \right\}^{2^{d_i} - 2} \right]$$
(3)

Hence
$$P(\text{all i.c.s and function are correctly identified}) = P_f P_{ic}$$
 (4)

where P_f is the probability of correct estimation of the function given that the i.c.s have been correctly determined, and is equal to the expression in (2)).

In order to give an idea of the cipher length requirements involved, we consider the estimation of the inputs for the case:

$$f = X_1 X_2 + X_3$$

where the LFSR of each input is of size 12. The values $p_1 = 0.52$, $p_2 = 0.52$, $p_3 = 0.56$ are computed from the truthtable. We set the overall probability of correct identification equal to 0.95. Using (3) we obtain a cipherlength of 20,225 bits for the identification of the 3 initial conditions. On the other hand, using (4), we obtain a cipherlength of 21,450 bits for the identification of the initial conditions as well as estimation of the functions. It is indeed interesting to note that the task of function estimation requires very little of additional cipherlength over that of the job of identification of the initial conditions alone.

5 Determination of Initial Conditions when the Function Is Correlation Immune

Correlation immunity of a function from an information theoretic viewpoint has been described by Siegenthaler in [8]. An equivalent definition, given in [10] is as follows: An m-variable function $Y = f(X_1, X_2, \dots, X_m)$ is lth order correlation immune iff

$$P(Y = X_{i_k} | X_{i_1} = 0, X_{i_2} = 0, \dots, X_{i_{k-1}} = 0) = 0.5,$$

where i_1, i_2, \ldots, i_k is a set of distinct indices between 1 and $m, 1 \leq k \leq l$.

5.1 Method

It follows the above definition of correlation immunity that for an m-variable, lth order correlation immune function, there is an input X_i and an index set $\{i_1, i_2, \ldots, i_{l+1}\}$ such that

$$P(Y = X_{i_{l+1}} | X_{i_1} = 0, X_{i_2} = 0, \dots, X_{i_l} = 0) \neq 0.5.$$

Further,

$$P(C = X_{i_{l+1}} | X_{i_1} = 0, X_{i_2} = 0, \dots, X_{i_l} = 0) \begin{cases} \neq 0.5 & \text{for correct i.c.,} \\ = 0.5 & \text{for wrong i.c.} \end{cases}$$
 (5)

Note that the conditioning on the *specific* combination $X_{i_1} = 0, X_{i_2} = 0, \dots, X_{i_l} = 0$ is done without loss of generality. It was shown by [10] that conditioning on any other combination of values of the same inputs would produce a probability away from 0.5 when the correct initial conditions have been chosen.

Based on this idea, we adopt the following approach for determining the initial conditions when the combining function is unknown and correlation immune of an unknown order:

- 1. For the *i*th input, compute empirically $P(C=X_i|X_j=0)=f_{ij},\ i\neq j,\ 1\leq i,j\leq m,$ for all possible initial condition pairs. If the maximum of $|f_{ij}-0.5|$ is reasonably large as well as sufficiently separated from the next (lower) value of $|f_{ij}-0.5|$, then it can be safely deduced that the corresponding initial conditions for *i* and *j* are the required initial conditions. Continue the procedure for all *i-j* combinations. Stop, if this results in the determination of all the initial conditions, otherwise proceed to Step 2.
- 2. For the *i*th input, compute empirically $P(C = X_i | X_j = 0, X_k = 0) = f_{ijk}$, $i \neq j \neq k, 1 \leq i, j, k \leq m$ for all possible input combinations triplets. As before, if the maximum of $|f_{ij} 0.5|$ is reasonably large and well separated from the rest, then the corresponding initial conditions are the correct ones. Continue this procedure for all i, j, k combinations. Stop, if all initial conditions have been determined. Otherwise, proceed to Step 3.
- 3. Condition on three inputs and proceed as before. If this fails, condition on four inputs, and so on.

5.2 Analysis

Let us use the notation $p_{i_{l+1}|i_1\cdots i_l}$ for $P(Y=X_{i_{l+1}}|X_{i_1}=0,X_{i_2}=0,\cdots,X_{i_l}=0)$, when the correct initial condition has been used. Note that the number of wrong i.c.s in this case is $\prod_{j=1}^{l+1}(2^{d_j}-2)$. Therefore, the probability of correct identification of the i.c.s of i_1,i_2,\ldots,i_{l+1} , after conditioning $X_{i_{l+1}}$ on $X_{i_1}=0,X_{i_2}=0,\cdots,X_{i_l}=0$), has an expression similar to that of P_{ic} in (3):

$$\sum_{u=0}^{n} \binom{n}{y} p_{i_{l+1}|i_1\cdots i_l}^y (1-p_{i_{l+1}|i_1\cdots i_l})^{n-y} \left\{ 2\Phi\left(\frac{|n-2y|}{\sqrt{n}}\right) - 1 \right\}^{\prod_{j=1}^{l+1} (2^{d_j}-2)},$$

where n is the number of bits (out of N) for which the input combination $X_{i_1} = 0, X_{i_2} = 0, \dots, X_{i_l} = 0$) actually occurs. It is clear that n has a Binomial distribution, $Bin(N, 2^{-l})$. Therefore, the probability of correct identification of the i.c.s of i_1, i_2, \dots, i_{l+1} , after conditioning $X_{i_{l+1}}$ on $X_{i_1} = 0, X_{i_2} = 0, \dots, X_{i_l} = 0$) is

$$P_{i_{l+1}|i_{1}\cdots i_{l}} = \sum_{n=0}^{N} {N \choose n} (2^{-ln}) (1 - 2^{-l})^{N-n} \sum_{y=0}^{n} {n \choose y} p_{i_{l+1}|i_{1}\cdots i_{l}}^{y} (1 - p_{i_{l+1}|i_{1}\cdots i_{l}})^{n-y}$$

$$\left\{ 2\Phi \left(\frac{|n-2y|}{\sqrt{n}} \right) - 1 \right\}^{\prod_{j=1}^{l+1} (2^{d_{j}} - 2)}.$$
 (6)

5.3 An Example

We illustrate the method for the function $X_1 + X_2 + X_4 + X_3X_5 + X_4X_5$, which is second-order correlation immune, and shift register sizes 4,5,6,7,8 respectively. The cipher length is taken as 24,000. Here, $P(C = X_1 | X_2, X_3 = 0) = P(C = X_4 | X_1, X_2 = 0) = .54$. The input X_5 is third order correlation immune, and $P(C = X_5 | X_1 = X_2 = X_3 = 0) = .46$. We start with the assumption that the function is unknown.

The empirical values of $P(C=X_i)$ are first calculated but none observed to be significantly away from 0.5, specifically, the maximum separation was 0.01. Next, the empirical values of $P(C=X_{i_k}|X_{i_j}=0)$ for all possible but distinct values of i and j are calculated. Once again, these are all close to 0.5, with a maximum deviation of 0.02. After this, the empirical values of $P(C=X_{i_k}|X_{i_j}=X_{i_l}=0)$ are calculated. It is observed that the empirical version of $P(C=X_1|X_2=X_3=0)$, has the largest separation (0.048) from 0.5 for the correct combination of i.c.s of X_1 , X_2 and X_3 , while the second highest separation is 0.037. Using the identified i.c.s of X_1 and X_2 , the empirical value of $P(C=X_4|X_1=X_2=0)$ is also found to be furthest away from 0.5, specifically 0.044, when the correct i.c. for X_4 is used. The input X_5 however, is found to have 'input immunity' of order 3 i.e., conditioning on two inputs does not yield a large enough value for the corresponding fraction of coincidence. (We define input immunity as follows: An input i has immunity of order m if $P(C=X_i|X_{i_1},X_{i_2},\ldots,X_{i_{m-1}}=0)=1/2$, $i \neq \{i_1,\ldots i_{m-1}\}$). It is found that

the empirical value of $P(C = X_5|X_1 = X_2 = X_3 = 0)$ has a well-defined separation from 0.5 (equal to 0.047), for the correct i.c. for X_5 . For every wrong initial condition of X_5 the separation from 0.5 is much less with a maximum equal to 0.037.

5.4 The 'Diameter' Approach

Note from (5) that the conditional probability of coincidence is only known to be 'away from 0.5'. The precise value of this probability may depend on the combination of values of the inputs on which the conditioning is made. Specifically, the 'all zero' combination may not produce a conditional probability furthest away from 0.5. In order to extract the maximum possible information from the conditioning process, we may try and condition on all possible combination of values of the conditioning inputs, and look for the maximum deviation from 0.5. Thus, for every input combination $X_{i_1}, X_{i_2}, \ldots, X_{i_{l+1}}$ we consider the absolute difference

$$\max_{\substack{u_1, u_2, \dots, u_l \text{ binary} \\ v_1, v_2, \dots, v_l \text{ binary}}} | P(C = X_{i_{l+1}} | X_{i_1} = u_1, X_{i_2} = u_2, \dots, X_{i_l} = u_l)$$

$$-P(C = X_{i_{l+1}} | X_{i_1} = v_1, X_{i_2} = v_2, \dots, X_{i_l} = v_l) |$$
(7)

which should be far away from 0 for the correct i.c. and equal to 0 for any wrong i.c. The empirical version of this maximum difference can be computed using the observed fraction of coincidences, and used in the procedure given in Section 5.3 in lieu of $|f_{ij}-0.5|$. The maximum absolute difference can be thought of as the diameter of the set of all candidate fractions. Therefore, we call this approach the 'diameter' approach.

When the diameter approach is used in the example of the above Section, it is found that the i.c.s for X_1 , X_2 and X_3 are correctly determined for a cipher length of only 16,000 bits. The corresponding fraction of coincidence (best) is 0.093 while the one immediately next to it in magnitude is 0.052. Thus, we see that the 'best' diameter is well separated from both 0 as well as the 'next best' diameter for even 16,000 bits. The rest of the search for the i.c.s of X_4 and X_5 are carried out successfully in a similar way. This suggests that a reasonable reduction in the cipher length requirements may be obtained using the 'diameter' approach.

6 Computational Work

The size of the space of test initial conditions grows exponentially with increase in the size of the LFSRs. For an input of size d_i which is not correlation immune, the search time is proportional to 2^{d_i} . If the input 1 has input immunity of order m and d_2, \ldots, d_m are corresponding conditioning inputs, then the search time is proportional to $2^{\sum d_i} = 2^{d_1+d_2+\cdots+d_m}$. The overall computational time is of the

order of the time needed for the determining the input with largest immunity and the corresponding conditioning inputs. Estimation of the combining function would have no effect on the order of the search time.

The actual software implementation of the LFSR simulation and output generation followed by comparison with the cipherstream is done based on certain efficient features proposed in [11]. To begin with, the cipherstream is 'packed' into an array. This is done by moving blocks of the ciphertext data (of length 16, in this paper) into each location of the array. For example if $\mathtt{ctext}[\]$ represents the packed ciphertext array and $\{c_0, c_1, c_2, \ldots, c_N\}$ is the ciphertext, then $\mathtt{ctext}[0]$ holds c_0, c_1, \ldots, c_{15} , $\mathtt{ctext}[1]$ holds $c_{16}, c_{17}, \ldots, c_{31}$ and so on. Next, the LFSR output is generated in packed form. The content of each packed location is compared with the content of the corresponding packed ciphertext array location by bitwise ex-oring. The number of bits for which the two do not match (the number of ones in the bit configuration of the resulting number) is read from a table-lookup. This number is obtained cumulatively for the entire packed ciphertext array. From this, the relevant fraction of coincidences is computed.

It was experimentally observed that for a LFSR of length 16, determination of the correct i.c. using no 'packing' required 985 seconds on a 333 MHz Pentium Processor. On the other hand, the algorithm incorporating data packing required only 89 seconds implying a significant speedup. Modifications for further increase in speed of the algorithm are currently being explored.

In order to have an idea of the actual running times involved, the algorithm with data packing was used on a 333 MHz Pentium processor. Determining the initial conditions for an input with input immunity $2,\{d_1=4,d_2=5,d_3=6\}$ required 900 seconds, $\{d_1=4,d_2=5,d_3=7\}$ required 1908 seconds while $\{d_1=5,d_2=6,d_3=7\}$ required 8429 seconds.

7 Conclusions

We have shown in Section 4 that the knowledge of the combining function is not a very important one, because the lack of this knowledge entails minimal increase in the cipherlength needed to break the code. The assumption of known shift register sizes and polynomials is easily removed if one permits a larger search space, *i.e.* the search must now include varying shift register sizes and polynomials. Other modifications such as parallelization must be explored to reduce the resulting computational workload.

In order to reduce the cipher length requirements one may modify the 'best-is-correct' approach to include a reasonable size of candidate solutions (k > 0). For example, the best 5% of the fractions of coincidence may be chosen and the corresponding i.c.s used to decrypt the message. The correct i.c.s will be the ones corresponding to which meaningful text (English or otherwise) is generated. This generation may even be automatised using some prominent features of the language. The trade-off between cipher length requirements and the computation needed for automatic checking of trial solutions should be an interesting subject of further study.

The primary bottleneck of the approach developed for correlation immune combining functions is the tremendous size of the search space for even low sizes of the LFSRs. The use of even fast modifications of the basic algorithm [11] do not contribute much to decreasing the computation time. Hence, in order to make such an encryption system cryptologically strong, the designer has to choose a combining function having a large number of inputs and correlation immunity of sufficiently high order (not too high because then, search by enumeration would be possible).

References

- A. Menezes, P. van Oorschot and S. Vanstone, Handbook of applied cryptography, CRC Press, 1997. 306
- T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only," IEEE
 Transactions on Computers, Vol. c-34, No.1, January 1985, pp. 81-85. 306, 307,
 308
- K. Zeng and M. Huang, "On the linear syndrome method in cryptanalysis," Advances in Cryptology CRYPTO '88, Vol. 403, S. Goldwasser, editor, Springer Verlag 1990, pp. 469-478. 307
- V. Chepyzhov and B. Smeets, "On a fast correlation attack on stream ciphers," Advances in Cryptology EUROCRYPT '91, Vol. 547, D.W. Davies, editor, Springer Verlag, 1991, pp. 176-185. 307
- A. Clark, J. Golić and E. Dawson, "A comparison of fast correlation attacks," Fast Software Encryption, Third International Workshop (LNCS 1039), D. Gollman, editor, Springer Verlag, 1996, pp. 145-157. 307
- R. Forre, "A fast correlation attack on non-linearly feedforward filtered shift register sequences," Advances in Cryptology EUROCRYPT '89 (LNCS 434), 1990, pp. 586-95.
- M.J. Mihaljevic and J. Golić, "A comparison of cryptanalytic principles based on iterative error-correction," Advances in Cryptology - EUROCRYPT '91, Vol. 547, D.W. Davies, editor, Springer Verlag 1991, pp. 527-531. 307
- T. Siegenthaler, "Correlation-Immunity of Nonlinear Combining functions for Cryptographic Applications," *IEEE Transactions on Information Theory*, Vol. 30, No. 5, September 1984, pp.776 - 780.
- B.K. Roy, "Ciphertext only cryptanalysis of LFSR based encryption schemes," Proceedings of the National Seminar on Cryptology, Delhi, July 1998, pp.A-19
 –A 24. 307, 316 308
- S. Maitra, P. Sarkar and B.K. Roy, "A new definition of correlation immunity of Boolean functions," *Technical Report No. ASD/98/08* Indian Statistical Institute, June 1998. 316
- S. Maitra and P. Sarkar, "Efficient implementation of ciphertext only attack on LFSR based encryption schemes," *Proceedings of the National Seminar on cryptology*, Delhi, July 1998, pp. A-1-A-12. 319, 320

Doing More with Fewer Bits

A.E Brouwer¹, R. Pellikaan¹, and E. R. Verheul²

 Department of Math. and Comp. Sc., P.O. Box 513, Eindhoven University of Technology, 5600 MB, Eindhoven, The Netherlands. [aeb,ruudp]@win.tue.nl
 PricewaterhouseCoopers GRMS Crypto Group P.O. Box 85096, 3508 AB Utrecht Eric.Verheul@[nl.pwcglobal.com, pobox.com]

Abstract. We present a variant of the Diffie-Hellman scheme in which the number of bits exchanged is one third of what is used in the classical Diffie-Hellman scheme, while the offered security against attacks known today is the same. We also give applications for this variant and conjecture a extension of this variant further reducing the size of sent information.

1 Introduction

In the classical Diffie-Hellman key-exchange scheme, two system parameters are fixed: a large prime number P and a generator g of the multiplicative group of the basic finite field GF(P). If two parties, Alice and Bob say, want to agree on a common secret key over an insecure channel, then Alice generates a random key $0 \le x < P-1$ and sends $A=g^x \mod P$ to Bob. Also, Bob generates a random key $0 \le y < P-1$ and sends $B=g^y \mod P$ to Alice. Both Alice and Bob can now determine the common, secret key $S=g^{xy} \mod P=A^y \mod P=B^x \mod P$. For adequate security, P should be a 1024 bit prime, such that P-1 contains a 160 bit prime factor (see below). In particular this means that all system parameters (i.e. g, P) and sent information (i.e. A, B) are of size 1024 bits.

In [14], Claus Schnorr proposed a variant of the classical Diffie-Hellman scheme, in which g does not generate the whole multiplicative group of the basic finite field GF(P), but only a small subgroup of which the order contains a 160 bit prime number q. As is suggested by Arjen Lenstra in [7], one can extend the Schnorr scheme to any multiplicative subgroup $G = \langle g \rangle$ of an extension field $GF(p^t)$. Lenstra specializes to generators g that have prime order, which we do as well.

For solving the discrete logarithm problem for a generator g of prime order q, one can use an index calculus (IC) based algorithm that has a heuristic expected asymptotic running time of $L(p^s, 1/3, 1.923 + o(1)]$, see [1] and [7], where s is the smallest divisor of t such that $\langle g \rangle$ is contained in a subfield of $GF(p^t)$ isomorphic to $GF(p^s)$. If p=2 then the constant 1.923 can be replaced by 1.587, see [3]. Alternatively one can use Birthday Paradox (BP) based algorithms (e.g. Pollard's rho algorithm [13]) that have expected running times exponential in the size of the q. More precisely, breaking the Discrete Logarithm problem can be solved in expected $O(\sqrt(q))$ elementary operations in $GF(p^t)$.

This leads us to the conclusion from [7] that - w.r.t. attacks known today - if the minimal surrounding subfield of g and prime order q of g are of sufficient size, then the discrete logarithm problem $\langle g \rangle$ is intractable. The particular form of the field itself, is not relevant. In other words, if $GF(p^t)$ is the minimal surrounding field of a subgroup of prime order, then - w.r.t. attack known today - the discrete logarithm in this subgroup is approximately as difficult as the discrete logarithm in a subgroup of prime order of a basic field GF(P) if the size of P is approximately equal to as t times the size of p, and the order of both subgroups are about the same size. Hence, a suitable generator g in a field extension $GF(p^t)$ should not be contained in one of the proper subfields and should have a suitably large prime order. In practice, the size of the (minimal) surrounding field should be a ≥ 1024 bit number, and the prime order of the element should at least be of size ≥ 160 bits.

In [7], Lenstra proposes a simple, practical method for the construction of a field extension and suitable generator. The idea is that one fixes the size of the prime number p and a number t such that p^t is "large" enough. Then one looks for a large prime factor q in the value of the cyclotomic polynomial $\phi_t(p)$. The latter can be done using trial divisions with the primes up to, say 10^5 ; any other reasonable bound or method will do. Finally, one constructs a generator g of order q, by looking for an element different from 1, such that $g^{(p^t-1)/q}=1$.

Using the above construction, the size of q is about $\varphi(t) \cdot |p|$ bits (where $\varphi(.)$ is Euler's totient function) which grows as least as fast as $p^{t/\log(\log(t))}$. The complexity of the BP based algorithms grows much faster, than the complexity of the IC based algorithms. So if the size of the surrounding field is large enough to resist the sub-exponential, IC based algorithms, then the order ϖ will usually be large enough "automatically" to resist the BP based algorithms as well.

In this paper we propose a variant of the Diffie-Hellman scheme, using a multiplicative group G of an extension field $GF(p^6)$ as indicated above. For adequate security, the size of $GF(p^6)$ is a ≥ 1024 bit number and the order of $G = \langle g \rangle$ is a ≥ 160 bit prime factor of $\phi_6(p)$. Our scheme has the following properties:

- 1. Breaking our scheme, means breaking the Diffie-Hellman scheme in G, which in (today's) practice is as least as difficult as breaking the classical Diffie-Hellman scheme w.r.t. a modulus of comparable size.
- 2. All sent information (i.e. the A, B mentioned above) is only one third of the normal size, i.e. 342 bits. This makes our variant of the Diffie-Hellman scheme more competitive with Elliptic curve cryptosystems.

Outline

In Section 2, mainly as an appetizer, we will present a description of the variant of the Diffie-Hellman scheme based on Lucas sequences. In this variant all sent information (i.e. the A, B above) is only one half of the usual size (i.e. 512 bits in practice). In Section 3 we present an improvement of this scheme, in which all sent information is only one third of the usual size. A discussion of some applications of our scheme appears in Section 4, and in Section 5 we discuss and conjecture extensions of our scheme. We summarize our results in Section 6.

2 A Different View of the LUCDIF Cryptosystem

Central in the construction of the LUCDIF variant of the Diffie-Hellman key-exchange scheme is a \geq 512-bit prime number p, such that p+1 contains a prime factor q of at least 160 bits. We now consider the field extension $GF(p^2)$, in which the multiplicative group is of order $p^2-1=(p+1)(p-1)$. So we can construct an element q of order q in $GF(p^2)$; the prime numbers p,q and the generator q are the parameters of the LUCDIF system.

Now, if Alice and Bob want to agree on a common secret key, then Alice generates a random key $0 \le x < q$, forms $A = g^x + g^{-x}$ (in the field $GF(p^2)$) and sends this to Bob. Similarly, Bob generates a random key $0 \le y < q$, forms $B = g^y + g^{-y}$ and sends this to Alice. We will now first show that both Alice and Bob can determine the common, secret key $S = g^{xy} + g^{-xy}$.

For Alice to determine this shared secret key, she will first retrieve g^y from B. If we denote g^y by U, then we have the following equation

$$B = U + U^{-1} \tag{1}$$

Equality (1) is a quadratic equation in U (namely $U^2 - B * U + 1 = 0$) with coefficients in GF(p) (see below). By using standard techniques (adjoining roots using the "abc-formula" in a symbolic way), Alice can find the two solutions $U_{1,2}$ in $GF(p^2)$ of this equation. All calculations in $GF(p^2)$ are symbolic and are effectively performed using operations in GF(p).

These two solutions correspond exactly with g^y, g^{-y} . However, Alice has no idea which is g^y and which is g^{-y} , but that does not matter as she can determine $U_1^x + U_2^x$ by using her random key x. This is equal to $S = g^{xy} + g^{-xy}$, independent of the choice of U_1 and U_2 . In a similar way, Bob can construct S from A. It is indicated in [2], that the above scheme coincides with the variant of the Diffie-Hellman key-exchange scheme that was proposed and analyzed by a series of authors; [15] (where the name 'LUCDIF' was proposed), [11], [10], [12] and [8]. We will now proceed with showing two important properties of the LUCDIF cryptosystem: reduced size of sent information and security.

2.1 Reduced Size of Sent Information of the LUCDIF Cryptosystem

For this property, we will show that both A and B are elements of the basic field GF(p) and can therefore be represented by |p| (e.g. 512) bits each. To this end, we first observe that any element h in the group generated by g, has the following property which we shall use often: $h^p = h^{-1}$. This follows as the order, q, of g (and therefore of h) divides p + 1. We now have

$$A^p = (g^x + g^{-x})^p = g^{xp} + g^{-xp} = g^{-x} + g^x = A,$$

which means that A is an element of GF(p) as we needed to show. It similarly follows that B is an element of GF(p) also.

It easily follows from the above discussion, that it suffices to take p, q and $g + g^{-1}$ as the system parameters instead of p, q and g. Hence, an additional advantage of the LUCDIF variant is that the size of the system parameters are also about halve the normal size. The same is realized in the setting of RSA in [6].

2.2 Security of the LUCDIF Cryptosystem

Concerning the security, we will show that if somebody (an oracle \mathcal{O}) can break the LUCDIF system (i.e. determining S on basis of A and B) then one can break the Diffie-Hellman problem in the group $\langle g \rangle$ generated by g.

This would conclude the discussion security of the LUCDIF system. Indeed, the order q of g is a divisor of $p+1=\phi_2(p)$ and we recall from the introduction that breaking the Diffie-Hellman scheme in $\langle g \rangle$ is - w.r.t. to attacks known today - is as infeasible as breaking the standard Diffie-Hellman scheme of a comparable size.

To this end, suppose $\alpha = g^x$, $\beta = g^y$ are given, then one can first construct $\alpha + \alpha^p$, $\beta + \beta^p$ and use this as input for $\mathcal O$ to determine $S_1 = g^{xy} + g^{xyp}$. By applying the same technique to α and $\beta \cdot g = g^{y+1}$ one can also determine $S_2 = g^{x(y+1)} + g^{x(y+1)p} = \alpha \cdot g^{xy} + \alpha^p \cdot g^{xyp}$. This means we have the following equation:

$$\begin{pmatrix} 1 & 1 \\ \alpha & \alpha^p \end{pmatrix} \cdot \begin{pmatrix} g^{xy} \\ g^{xyp} \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix} \tag{2}$$

By this equation one now can deduce g^{xy} , i.e. the Diffie-Hellman shared secret key. Observe that the matrix above is regular because $\alpha \neq \alpha^p$ as α is not a member of GF(p) by construction. Conversely, if somebody can break the Diffie-Hellman scheme in $\langle g \rangle$, then it is simple to show that one can break the LUCDIF system.

3 Our System

What is done in the LUCDIF scheme from an algebraic point of view, is representing an element z of the extension field $GF(p^2)$ not as the usual residue class modulo a fixed irreducible polynomial of degree 2, but by its unique, minimal polynomial, see [9]. If the element z is chosen in $\langle g \rangle$, then we can save information as the constant term of the minimal polynomial is always one, leaving only the first order coefficient (an element of GF(p)) to be stored or sent. However, the minimal polynomial of z does not only represent z, but also its conjugate $z^p = z^{-1}$. That is why we take the sum of the two conjugates (a symmetric function) to represent the exchanged key in the Diffie-Hellman scheme. We have shown that this is no problem with respect to security.

In this section we will develop a generalization of this technique in $GF(p^6)$. Central in our generalization is a 171-bit prime number p, such that the sixth cyclotomic polynomial $\phi_6(p) = p^2 - p + 1$ (see [9]) contains a prime factor q of at least 160 bits. As $\phi_6(p)$ is a divisor of $p^6 - 1$ (the order of the multiplicative

group of $GF(p^6)$) we can easily construct a generator g in $GF(p^6)^*$ of order q. The prime numbers p, q, and the generator g are the parameters of our system. Actually, by taking a different representation of g the size of the system parameters can be reduced, see subsection 3.1.

Now, if Alice and Bob want to agree on a common secret key, then Alice generates a random key $0 \le x < q$, and forms the minimal polynomial P(X) of g^x , i.e.

$$P_A(X) = \prod_{i=0}^{5} (X - g^{xp^i}) = X^6 + A_5 X^5 + A_4 X^4 + A_3 X^3 + A_2 X^2 + A_1 X + 1,$$

where all A_i are in GF(p). Note that the constant term of $P_A(X)$ is 1 as $\phi_6(p)$ divides $1 + p + ... + p^5 = (p^6 - 1)/(p - 1)$. Then, Alice sends (A_1, A_2) to Bob. Bob also generates a random key $0 \le y < q$, and forms the minimal polynomial P(X) of g^y , i.e.

$$P_B(X) = \prod_{i=0}^{5} (X - g^{yp^i}) = X^6 + B_5 X^5 + B_4 X^4 + B_3 X^3 + B_2 X^2 + B_1 X + 1$$
 (3)

where all B_i are in GF(p). Bob then sends (B_1, B_2) to Alice.

The shared secret key will be the pair (C_1, C_2) , i.e. the first and second order coefficients of the polynomial $P_C(X)$ given by:

$$P_C(X) = \prod_{i=0}^{5} (X - g^{xyp^i}) = X^6 + C_5 X^5 + C_4 X^4 + C_3 X^3 + C_2 X^2 + C_1 X + 1$$

For Alice to determine this shared secret key, she will first retrieve the polynomial $P_B(X)$ from (B_1, B_2) . To do this, she needs to reconstruct the B_3, B_4, B_5 from B_1, B_2 . To this end, as $p^3 = -1 \mod p^2 - p + 1$ it follows that the polynomial $P_B(X)$ is symmetric, i.e. $B_5 = B_1$ and $B_4 = B_2$.

It also follows that if we denote g^y by β , and $\beta_i = \beta^{p^i}$ for i = 0, 1, 2, 3, 4, 5 then

$$\beta_0 = \beta, \ \beta_1 = \beta^p, \ \beta_2 = \beta^{p-1}, \ \beta_3 = \beta^{-1}, \ \beta_4 = \beta^{-p}, \ \beta_5 = \beta^{1-p}.$$
 (4)

By equation (3) one can write B_3 in terms of the $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5$, and by using the reductions from (4), one can easily verify that:

$$B_3 = -2 \cdot \sum_{i=0}^{5} \beta_i - \sum_{i=0}^{5} \beta_i^2 - 2,$$

which is a symmetric polynomial in $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ of degree 2, and which can hence be written in the first and the second elementary polynomials of $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ by the so-called Newton equalities. Of course the value of the first symmetric polynomial equals $-B_1$ and the value of the second symmetric polynomial equals B_2 . This leads to the following, easily verified formula.

$$B_3 = -2 + 2 * B_1 - B_1^2 + 2 * B_2.$$

So, starting from (B_1, B_2) , Alice is able to retrieve the polynomial $P_B(X)$. She can then adjoin a root ρ of this polynomial to obtain $GF(p^6)$, next she can use her secret key x, to determine the minimal polynomial of ρ^x , which is equal to $P_C(X)$. That is, Alice is able to determine the shared secret key (C_1, C_2) . Similarly, Bob is able to determine (C_1, C_2) from (A_1, A_2) and his secret key y.

3.1 Reduced Size Property of Our System

Alice and Bob only send two coefficients in GF(p) to each other, which corresponds to only 2|p| bits of data.

It easily follows from the above discussion, that it suffices to take p,q and the first and second order coefficients (elements of GF(p)) of the minimal polynomial of g, as system parameters. Hence, a typical size of the system parameters of our scheme would be 673 bits, consisting of 171+160=331 bits for representing p and q plus 342 bits for representing g. The system parameters of a comparative Diffie-Hellman scheme would be 2048 bits (even 2208 bits for the Schnorr variant), which is more than three times as large.

3.2 Security of Our System

We'll first show that the security of our variant of the Diffie-Hellman scheme is equivalent to the security of the Diffie-Hellman scheme in $\langle g \rangle$. This would conclude the discussion security of our system, as we recall that from the introduction that breaking the Diffie-Hellman scheme in $\langle g \rangle$ is - w.r.t. to attacks known today - is as infeasible as breaking the standard Diffie-Hellman scheme of a comparable size.

To this end, consider the following two functions $Z_1(.), Z_2(.): \langle g \rangle \to GF(p^6)$ defined by:

$$Z_1(h) = \sum_{i=0}^{5} h^{p^i},$$

$$Z_2(h) = \sum_{0 < i \neq j < 5} h^{p^i + p^j}$$

Then, in the terminology of the previous section,

$$A_1 = Z_1(g^x), B_1 = Z_1(g^y), C_1 = Z_1(g^{xy})$$

 $A_2 = Z_2(g^x), B_2 = Z_2(g^y), C_2 = Z_2(g^{xy})$

We have the following result, the proof of which is similar to the argument used in proving the security of the LUCDIF system in Section 2.2

Lemma 3.1 For i = 1, 2, the problem of determining $Z_i(g^{xy})$ from $Z_i(g^x), Z_i(g^y)$ is as least as difficult as solving the Diffie-Hellman problem in $\langle g \rangle$.

The security of our variant of the Diffie-Hellman scheme is equivalent to the difficulty of determining $Z_1(g^{xy}), Z_2(g^{xy})$ from,

$$Z_1(g^x), Z_1(g^y), Z_2(g^x), Z_2(g^y).$$

Hence, it immediately follows from Lemma 3.1 that solving this problem is equivalent with solving the Diffie-Hellman problem in $\langle g \rangle$. With respect to the attacks known today, this is just as difficult as breaking the Diffie-Hellman scheme in a basic field GF(P) of size |P| = 6 * |p|.

So as we only need to exchange two coefficients of |p| bits each, all sent information is one third of the size of the standard Diffie-Hellman scheme, while the offered security is the same.

It is shown in [16] that Lemma 3.1 is a consequence of a much broader result. To this end, let t > 1 be an integer (t = 6 in the current setting). Let n be a non-negative number and consider the integers e_1 , e_n (the "exponents") and the elements $\lambda_1, ..., \lambda_n \in GF(p^t) \setminus \{0\}$ (the "multipliers") and consider the following summing function Z(.): $\langle g \rangle \to GF(p^t)$ defined by:

$$Z(\kappa) = \sum_{i=1}^{n} \lambda_i \cdot \kappa^{e_i}, \text{ for } \kappa \in \langle g \rangle$$

The number n is called the *degree* of the summing function and the number $d = \gcd(e_1, e_2, ..., e_n, \operatorname{ord}(g))$ is called the *order* of the summing function. It is a simple verification, that the above introduced $Z_1(.), Z_2(.)$ are summing functions of order 1.

Now the following "hardness" result is shown in [16].

Theorem 3.2 In the above terminology, let Z(.) be a summing function of order d. Also let \mathcal{O} be an oracle that on basis of any γ^x and γ^y computes $Z(\gamma^{xy})$. Then there exists a polynomial time algorithm that computes γ^{xyd} on basis of γ^x and γ^y . That is, for d=1 there exists a polynomial time algorithm that solves the whole Diffie-Hellman problem in $\langle \gamma \rangle$.

3.3 Implementation of Our System

In determining a shared secret key in our system a participant typically has to perform the following operations:

- 1. Restore the minimal polynomial he received, and adjoin a root ρ to GF(p) satisfying this polynomial, giving a copy of the field $GF(p^6)$.
- 2. Determining ϕ , i.e. ρ raised to his random key in the representation given by the root ρ and its minimal polynomial, i.e. as a linear combination of ρ^i , with i = 0, 1, 2, 3, 4, 5.
- 3. Determine the values of the first and second order coefficients of the minimal polynomial of ϕ .

The first operation is of negligible complexity. For the second operation a repeated square-and-multiply algorithm should be used, taking $O(36 \cdot \ln(q) \cdot \ln(p)^2)$ bit operations. For the final step, the representations of the conjugates of ϕ , i.e. ϕ^{p^i} for i = 1, 2, 3, 4, 5, have to be determined. As $p^3 = -1 \mod p^2 - p + 1$, only ϕ^p , ϕ^{p^2} need to be calculated. As furthermore

$$\phi^{p^2-p+1} = \phi^{p^2} \cdot \phi^{-p} \cdot \phi = 1$$

it follows that only the representation of ϕ^p needs to be calculated with a repeated square-and-multiply algorithm; the representations of the remaining conjugates can be determined by taking inverses. This takes an additional $O(36 \cdot \ln(q) \cdot \ln(p)^2)$ bit operations. The values of the first and second order coefficients of the minimal polynomial of ϕ can now be easily determined as the values of the first and second elementary symmetric polynomials of the conjugates of ρ . Given the representation of these conjugates, this is of negligible complexity. All and all, determining a shared secret key in our system by a participant takes $O(36 \cdot \ln(q) \cdot \ln(p)^2)$ bit operations, which is at least asymptotically comparable with the number of operations needed for a Diffie-Hellman key-exchange in a basic field of comparable size.

It's rather unfortunate that it seems that we can not fix the representation of $GF(p^6)$ to the one proposed by Arjen Lenstra in [7], in which an optimal normal basis can be used, such that exponentiation can carried out even more efficiently than in a basic field of the same size.

4 Applications of Our System

Our system can not only be used to obtain a variant of the Diffie-Hellman scheme where the exchanged data is one third of the usual size, but it can also be used to construct variants of schemes which are related to this scheme such as the ElGamal encryption scheme [4]. In this variant Bob's public key takes the form $(Z_1(y), Z_2(y))$, where $y = g^x$ and where $0 \le x < q$ is Bob's private key. An encryption for Bob (e.g. by Alice) of an element S in GF(p) takes the form

$$[(Z_1(g^k), Z_2(g^k)), S + Z_1(y^k)]$$

where k is taken randomly less than q. It follows from the discussion of the previous section, that Alice is indeed capable to determine this encryption, and Bob is indeed capable to decrypt this and recover S. It easily follows that the ability to break this system, means the ability to solve the Diffie-Hellman system in $\langle g \rangle$, which is, as far as is known today, just as difficult as breaking the Diffie-Hellman scheme in a basic field GF(P) of size |P| = 6 * |p|.

Of course, many other variants are possible, such as using $Z_2(.)$ instead of $Z_1(.)$. In this variant the size of the public keys (342 bits) is one third of the "normal" size. Moreover, the total encryption of an element of GF(p) takes a total size of 3*171 = 513 bits. So, when using hybrid encryption (only encrypting a random session key asymmetrically) for a non-interactive application (e.g. email), the

data requirement for our ElGamal variant is even less than for an RSA encryption of the same security. Actually, as in such cases only a random session key is required, one can use $Z_1(y^k)$ (with k random) as the session key, and include $(Z_1(g^k), Z_2(g^k))$, with the message. In this fashion, only one third of the size is required for the asymmetric part of what is usually used for an RSA encryption.

Apart from asymmetric schemes for confidentiality, the scheme can be used to make variants of digital signature schemes like the Digital Signature Algorithm [5], with public keys that are only a third of the usual size. The idea is that a verification of type $v = g^{u_1} \cdot y^{u_2}$, occurring in the verification part of the Digital Signature Algorithm where v, u_1, u_2 are constructed from the digital signature, can be recovered from $(Z_1(g), Z_2(g))$ and $(Z_1(y), Z_2(y))$, albeit not in a unique fashion: there are $6 \cdot 6 = 36$ possibilities, leading to 36 verifications one of which should hold. Of course, several straight-forward techniques can be employed to reduce this number of verifications. For instance, at the cost of a one time distribution of the (whole) generator g, instead of its minimal polynomial, one can reduce the number of verifications to six.

5 Extensions of Our System

Consider a prime number p and an integer t such that:

- 1. the multiplicative group of $GF(p^t)$ is large enough to withstand the Index Calculus based attacks;
- 2. the t-th cyclotomic polynomial $\phi_t(p)$ contains a prime factor q that is large enough to withstand the Birthday Paradox based algorithms.

Next a generator g is chosen of order q. Any element $h \in \langle g \rangle$ can be represented as an element of $GF(p^t)$ using $t \cdot |p|$ bits. However, the degree of the t-th cyclotomic polynomial is equal to $\varphi(t)$, where $\varphi(.)$ is Euler's totient function. So, in principle only $\varphi(t) \cdot |p|$ bits are required to represent any element $h \in \langle g \rangle$. A straightforward way to do this is to write $h = g^x$ for some $0 \le x < q$ and to represent h by x. However, this means solving the discrete logarithm problem for h with respect to g. What we aim for, is a technique to represent ("compress") h by only $\varphi(t) \cdot |p|$ bits, without solving the discrete logarithm problem for h with respect to g. With this technique, Alice and Bob can agree on a common secret key in $\langle g \rangle$ by sending each other only $\varphi(n) \cdot |p|$ bits.

Motivated by the techniques from Sections 2 and 3 we conjecture the following "compressed" form of the Diffie-Hellman scheme.

If Alice and Bob want to agree on a common secret key, then Alice generates a random key $0 \le x < q$, and forms the minimal polynomial $P_A(X)$ of g^x and somehow represents this with using only $\varphi(t) \cdot |p|$ bits and sends this to Bob. Similarly, Bob generates a random key $0 \le y < q$, and forms the minimal polynomial $P_B(X)$ of g^y and somehow represents this with using only $\varphi(t) \cdot |p|$ bits and sends this to Alice. Using this representation, Alice is able to determine

the minimal polynomial $P_B(X)$, and to adjoin a root ρ of $P_B(X)$ to GF(p) and to determine a representation of $GF(p^t)$. Using this she is able to determine the minimal polynomial $P_C(X)$ in GF(p) of ρ^x , which is equal to the minimal polynomial of g^{xy} . Alice uses the coefficient S of the first order term in $P_C(X)$ (i.e. the sum all conjugates of g^{xy}) as a secret key. Similarly, Bob is able to determine S using the representation of $P_A(X)$ and his private key.

With such a scheme, one obtains a version of the Diffie-Hellman scheme in which one only sends a $\frac{\varphi(t)}{t}$ part of the information one would normally send. If t is the product of the first k prime numbers, then one can easily show that this fraction goes to zero if k goes to infinity. So the achieved reduction gets better all the time. Also, using Theorem 3.2 one can show that (given such a representation) breaking this scheme, means solving the Diffie-Hellman problem in $\langle g \rangle$, which is suitable intractable, as far as is known today. Moreover, several other, secure choices (values of a symmetric function in all conjugates of g^{xy}) for the shared secret key S are possible.

Of course, the problem is how to represent such minimal polynomials by only $\varphi(t) \cdot |p|$ bits. In Sections 2 and 3 we have given such representations for t=2 and t=6. If t is a prime number r, then such a representation is straightforward. Indeed, as $\phi_t(p)$ divides $(p^t-1)/(p-1)$, the constant term of the minimal polynomials is 1. Hence only $r-1=\varphi(r)$ coefficients are unknown, each of size |p| bits.

Moreover, the representation used for t=6 can be extended to the case where t is of type 2r where r is prime as follows. Let, as before, $P_A(X)$ be the minimal polynomial of degree t of g^x in GF(p). Then, $P_A(X)$ splits as a product of two polynomials $P_{A_1}(X)$, $P_{A_2}(X)$ of degree r in $GF(p^2)$. Let us denote:

$$P_{A_1}(X) = X^r + a_{r-1,1}X^{r-1} + \dots + a_{1,1}X + 1$$
(5)

$$P_{A_2}(X) = X^r + a_{r-1,2}X^{r-1} + \dots + a_{1,2}X + 1.$$
(6)

The constant terms of both polynomials are 1 as $\phi_{2r}(p)$ divides $(p^{2r}-1)/(p^2-1)$. Then it easily follows that

$$(a_{i,1})^p = a_{i,2}$$
 for all $i = r - 1, ..., 2.$ (7)

It also follows that the reciprocal polynomial of $P_{A_1}(X)$ coincides with $P_{A_2}(X)$, i.e.:

$$X^r \cdot P_{A_1}(1/X) = P_{A_2}(X). \tag{8}$$

Now, suppose one possesses the first (r-1)/2 non-trivial coefficients of $P_{A_1}(X)$, i.e. $a_{r-1,1},...,a_{(r-1)/2-1,1}$, then using formula (7) one obtains the first (r-1)/2 non-trivial coefficients of $P_{A_2}(X)$, i.e. $a_{(r-1)/2,2},...,a_{(r-1)/2-1,2}$. From these and formula (8) it follows that one obtains the remaining (r-1)/2 non-trivial coefficients of $P_{A_1}(X)$, i.e. $a_{(r-1)/2,1},...,a_{1,1}$.

That is, from the first (r-1)/2 non-trivial coefficients of $P_{A_1}(X)$ one can reconstruct $P_{A_1}(X)$, and hence $P_A(X)$.

Hence, we can represent $P_A(X)$ by (r-1)/2 coefficients in $GF(p^2)$ for which one only needs $(r-1)/2 \cdot 2|p| = (r-1)|p| =$ bits, which is in accordance with our conjecture.

Problem How to find representations of minimal polynomials of only $\varphi(t) \cdot |p|$ bits for general t?

Irrespective of their existence, we note that the number of extensions of our system, as discussed above, that are more efficient in practice is quite low. To illustrate, assuming that for the coming years a classical (e.g. RSA) asymmetric key length between 1024 and 2048 bits gives adequate security for most (commercial) applications, then there are actually only two possible more efficient practical extensions. The first one, corresponds with t equal to $30 = 2 \cdot 3 \cdot 5$, where a reduction of $\varphi(30)/30 = 4/15$ can then be achieved. The characteristic of the used field would be a prime number between 35 and 70 bits length. The second one, corresponds with t equal to $210 = 2 \cdot 3 \cdot 5 \cdot 7$, where a reduction of $\varphi(210)/210 = 8/35$ can then be achieved. The characteristic of the used field would be a prime number between 5 and 10 bits length. For t equal to $2310 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11$, the used field size would have to larger than a 2048 bit number.

6 Conclusion

We have presented a variant of the Diffie-Hellman scheme in which all sent information is one third of the size of the standard Diffie-Hellman scheme, while the offered security, as far as is known today, is the same. We have also given applications for this construction. Finally, we have given a conjecture for an extension of our scheme in which all sent information is only a factor $\varphi(t)/t$ of the size of the standard Diffie-Hellman scheme.

References

- M. Adleman, J. DeMarrais, A subexponentional algorithm over all finite fields, CRYPTO '93 Proceedings, Springer-Verlag, pp. 147-158.
- D. Bleichenbacher, W. Bosma, A.K. Lenstra, Some remarks on Lucas-Based Cryptosystems, CRYPTO '95 Proceedings, Springer-Verlag, pp. 386-396. 323
- D. Coppersmith, Fast evaluation of logarithms in fields of characteristic two, IEEE Transactions on Information Theory, 30, (1984), pp. 587-594.
- T. ElGamal, A Public Key Cryptosystem and a Signature scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory 31(4), 1985, pp. 469-472. 328
- FIPS 186, Digital signature standard, Federal Information Processing Standards Publication 186, U.S. Department of Commerce/ NIST, 1994.
- A.K. Lenstra, Generating RSA moduli with a predetermined portion, Asiacrypt '98 proceedings, Springer-Verlag, pp. 1-10.

- A.K. Lenstra, Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields, Information Security and Privacy ACISP97 Proceedings (Sydney 1997), Lect. Notes in Comp. Sci. 1270, Springer-Verlag, pp. 127-138. 321, 321, 322, 322, 328
- 8. R. Lidl, W.B. Müller, *Permutation Polynomials in RSA-cryptosystems*, Crypto '83 Proceedings, Plemium Press, pp. 293-301. 323
- 9. R. Lidl, H. Niederreiter, Finite Fields, Addison-Wesley, 1983. 324, 324
- W.B. Müller, Polynomial functions in modern cryptology, Contributions to general Algebra 3, Proceedings of the Vienna Conference (1985), pp. 7-32. Proceedings, Springer-Verlag, pp. 50-61. 323
- W.B. Müller, W. Nöbauer, Cryptanalysis of the Dickson-Scheme, Eurocrypt '85 Proceedings, Springer-Verlag, pp. 50-61. 323
- 12. W. Nöbauer, *Cryptanalysis of the Rédei Scheme*, Contributions to general Algebra 3, Proceedings of the Vienna Conference (1985), pp. 255-264. 323
- J.M. Pollard, Monte Carlo methods for index computation (modp), Mathematics of Computation, 32, (1978), pp. 918-924.
- C.P. Schnorr, Efficient signature generation by smart cards, Journal of Cryptology, 4, pp. 161-174 (1991).
- 15. P. Smith, C. Skinner, A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms, Asiacrypt '94 proceedings, Springer-Verlag, pp. 357-364. 323
- E.R. Verheul, Certificates of Recoverability with Scalable Recovery Agent Security, in preparation. 327, 327

A Quick Group Key Distribution Scheme with "Entity Revocation"

Jun Anzai¹, Natsume Matsuzaki¹, and Tsutomu Matsumoto²

Abstract: This paper proposes a group key distribution scheme with an "entity revocation", which renews a group key of all the entities except one (or more) specific entity (ies). In broadcast systems such as Pay-TV, Internet multicast and mobile telecommunication for a group, a manager should revoke a dishonest entity or an unauthorized terminal as soon as possible to protect the secrecy of the group communication. However, it takes a long time for the "entity revocation" on a large group, if the manager distributes a group key to each entity except the revoked one. A recently published paper proposed a group key distribution scheme in which the amount of transmission and the delay do not rely on the number of entities of the group, using a type of secret sharing technique. This paper devises a novel key distribution scheme with "entity revocation" that makes frequent key distribution a practical reality. This scheme uses a technique similar to "threshold cryptosystems" and the one-pass Diffie-Hellman key exchange scheme.

1 Introduction

It is required a secure and quick key distribution scheme suitable for such broadcast systems which dynamically change the compose of the group, as Pay-TV (broadcasting via satellite or via cable), Internet multicasts (push, streaming or conference systems, for example) and mobile telecommunication for a group (private mobile radio or taxi radio, for example).

In this paper, we focus on a group key distribution scheme for all the entities except one (or some) specific one(s), that is called "entity revocation" here. "Entity revocation" is an essential mechanism for a secure group communication, if we consider the situation when an unauthorized user might eavesdrop using a lost or stolen terminal of mobile telecommunications or when an entity that left off from a conference system has continued to hear the secret communication of the conference. Also "entity revocation" is necessary to prevent dishonest entities from enjoying a pay service like Pay-TV and pay Internet without paying a charge.

A Familiar method for "entity revocation", called "Familiar method" in this paper, is that a key distributor distributes a new group key to each entity except the revoked

¹ Advanced Mobile Telecommunications Security Technology Research Laboratories 3-20-8, Shinyokohama, Kohoku-ku, Yokohama, Kanagawa, 222-0033 Japan {anzai, matuzaki }@amsl.co.jp

² Division of Artificial Environment and Systems, Yokohama National University, 79-5, Tokiwadai, Hodogaya, Yokohama,240-8501 Japan

entities, as encrypted form by a secret key of each entity. However, the amount of transmission and the delay become large when the group becomes large.

Papers [7] and [8] proposed a concept and a concrete scheme of a conference key distribution for secure digital mobile communications with low computational complexity. Also paper [2] proposed a method to expand Diffie-Hellman key exchange scheme so as to share a key among three or more entities through broadcast network. Since their schemes basically involve a key distribution for the other entities except the revoked one, the feature is the same as "Familiar method". So, one of our goal is to propose a scheme with "entity revocation", in which the amount of transmission and the delay do not rely on the group scale and analyze its security and performance.

On the other hand, a scheme proposed in paper [9] enables entities to share a key so that the amount of transmission and the delay do not rely on the group scale. The purpose of this scheme is that a data supplier can trace malicious authorized users who gave a decryption key to an unauthorized use. However, this scheme can not be applied to "entity revocation".

Recently, paper [10] has proposed two methods which enable an efficient "entity revocation" of which the amount of transmission and the delay don't rely on the group scale. However, the methods require a preparation phase when a distributor sends each encrypted group key for each entity before deciding a revoked entity. Therefore, the methods are not suitable for a system of which "entity revocation" happens frequently. Moreover, the methods require a fixed and privilege distributor who manages all secret keys of other entities.

In this paper, we propose a group key distribution scheme with "entity revocation" to achieve the following requirements:

- -The amount of transmission and the delay, from deciding a revoked entity until completing a group key distribution for all entities, does not rely on the group scale. We believe this requirement is effective to achieve a quick key distribution with "entity revocation" when the group is large.
- -Preparation phase when a distributor sends each encrypted group key to the corresponding entity, is not necessary. This requirement is suitable for a system with a frequent "entity revocation".
- -The fixed-privileged distributor isn't required. Anyone, called "coordinator" in this paper, can do "entity revocation".

The organization of this paper is as follows. In Section 2, we will explain our approach by examining two methods in paper [10]. In Section 3, we propose our scheme to satisfy the above requirements after explaining our target system. In Section 4, we discuss the security of our proposed scheme. In Section 5, we describe some considerations that are necessary to apply our scheme to an actual system. Also, in Section 6, we evaluate the performance and features of our scheme.

2 Approach

In this section, we investigate basic methods seen in paper [10] to fulfill which satisfies a part of our requirements. We modify it into a simpler method because the methods shown in [10] are rather complicated to analyze. And we pick up the above mentioned remaining problems and show our approach to solve them.

2.1 Basic Methods from Paper [10]

The methods shown in paper [10] satisfy the first requirement that the amount of transmission and the delay do not rely on the group scale.

The methods consist of two steps. In the first step, a fixed-privileged distributor generates a group key and sends the encrypted group key called "preparation data" here, for each entity, in such a way that any entity has not been able to decrypt it yet. In the second step, the distributor decides which entity should be revoked and broadcasts the secret key of the revoked entity. The amount of transmission and the delay in the second step do not rely on the number of entities. Receiving the broadcast data, all the entities except the revoked one can decrypt the preparation data to get the group key. The methods use a mathematical technique that is known as "RSA common modulus attack" [11] and "RSA low exponent attack" [6] respectively to realize:

- -to distinguish a revoked entity from the other entities, and
- -to share a same group key among the other entities.

Here we call the method by using technique of [11] "Previous Method 1", the method by using technique of [6] "Previous method 2".

We think their attacks are a type of "secret sharing schemes". However, "RSA low exponent attack" uses Chinese remainder theorem, similar to a secret sharing scheme proposed in paper [1].

Next, we show a modification of the scheme in paper [10], using a general secret sharing technique.

2.2 A Modification of The Scheme from Paper [10]

- 1 We assume there exists a secure communication path between a distributor and each entity, using symmetric cryptography or asymmetric cryptography.
- 2 The distributor generates a secret data S as the group key, and divides it by threshold 2, using the general secret sharing technique shown in paper [12]. And the distributor sends each shadow s_i to entity i as its secret key, through each secure communication path of step1. So it takes a time relying on the group scale to distribute all the shadows.

- 3 Let us support the distributor needs to revoke the entity j. Then the distributor broadcasts the secret key s_j of entity j. Of course, this amount of transmission and the delay don't rely on the group scale.
- 4 Receiving the secret key s_j , all entities except the revoked one can recover the group key S by using two sets of shadow: its own secret key and the secret key s_j . The revoked entity alone can't recover the group key S because it can get only one secret key s_i .

This scheme can expand so as to revoke k-1 entities at a time, by dividing the secret S by threshold k.

2.3 Our Approach

In both methods seen in paper [10] and in the modified scheme shown above, it is necessary for the distributor to send each preparation data (or shadow) to the corresponding entity. Therefore, it takes a long time to finish distributing the preparation data when the group is large. So we consider that these schemes unsuitable for a system with frequent "entity revocation". We need a scheme that reuses the distributed shadow while maintaining high security high. Also, because the methods in paper [10] use an RSA-like cryptosystem, heavy calculation with long integers is required for each entity. So we need a scheme of which security is based on discrete logarithm problem (DLP), in order to reduce data size and calculation time while maintaining high security, by using elliptic curve cryptosystems (ECC) or hyper-elliptic curve cryptosystems (HCC).

Also, the method in paper [10] and the modified scheme require a fixed-privileged distributor to manage the secret keys of all the entities. We require a scheme whereby any entity can become a distributor in order to apply it to a system where all members have equal rights, like a conference system.

Our approach to achieve the above requirements is as follows:

- -We apply "threshold cryptosystems" shown in paper [4] based on DLP to our purpose. This is a type of secret sharing scheme such that the entity can reuse the shadow. According this approach, we expect that the preparation phase is not needed and that the calculation time can be reduced by using ECC or HCC.
- -We use the one-pass Diffie-Hellman key exchange scheme to distribute preparation data. According this approach, we expect that any entity can become a distributor.

Now, we explain our scheme under above approaches.

3 Proposed Scheme

3.1 Target System

The broadcast system of our target is defined as follows:

System manager:

A trusted party who decides system parameters and sets each entity's secret key. Also it manages a public bulletin board.

Entity: i

A user or terminal that is a member of the group. We assume the group has n entities, and let Φ be a set of the entities:

$$\Phi = \{1, 2, ..., n\}.$$

Also, we assume that all entities are connected to a broadcast network and that any entity can send data to any other entities simultaneously.

Coordinator: v

A coordinator decides a (or some) revoked entity (ies) and coordinates a group key distribution with "entity revocation". We use the term "coordinator" to distinguish it from the fixed-privileged distributor discussed earlier. In our scheme, any entity can become coordinator.

Revoked entity: j

An entity to be revoked by the coordinator. Let Λ ($\subset \Phi$) be a set of revoked entities, having d entities.

Public bulletin board:

It keeps system parameters and public keys for all entities with certifications made by the system manager. We assume that any entity can get any data from this board at any time.

Thereafter, we explain our scheme, dividing into system setup phase and key distribution phase.

3.2 System Setup Phase

A system manager decides a parameter k that is satisfied:

$$0 \le d \le k-1 < n$$
,

where n is the number of entities in the group and d is the number of revoked entities.

1 The system manager decides the following system parameters and publishes them to a public bulletin board:

p: a large prime such that p > n+k-1 (about 1024 bit),

q: a large prime such that $q \mid p-1$ (about 160 bit) and

g: a q th root of unity over GF(p).

The system manager generates a system secret key $S \in \mathbb{Z}_q$, and stores it secretly.

2 The system manager divides the system secret key S into n+k-1 shadows by threshold k, using well-known Shamir's secret sharing scheme [12]:

 $1 a_0 = S$.

2 The system manager defines the following equation over GF (p):

$$f(x) = \sum a_f x^f \mod q, \tag{1}$$

$$f = 0$$

where $a_1, a_2, ..., a_{k-1}$ are random integers which satisfy the following conditions:

$$0 \le a_i \le q-1$$
 for all $1 \le i \le k-1$ and $a_{k-1} \ne 0$.

3 The system manager generates n+k-1 shadows as follows:

$$s_i = f(i) \ (1 \le i \le n+k-1).$$

- 3 The system manager distributes the shadows s_1 , ..., s_n to each entity 1, ..., n respectively through a secure way. Each entity keeps its own shadow as its secret key. The remaining k-1 shadows are safely stored as spare secret keys.
- 4 The system manager calculates public keys $y_1, ..., y_{n+k-1}$ by the following equation:

$$y_i = g^{s_i} \mod p \ (1 \le i \le n + k - 1).$$
 (2)

Then the system manager publishes y_1 , ..., y_n on the public bulletin board with the corresponding entity's identity numbers. The remaining y_{n+1} , ..., y_{n+k-1} are published to the public bulletin board as spare public keys.

3.3 Key Distribution Phase

<Generation of broadcast data by the coordinator>

First, a coordinator generates a broadcast data B (Λ, r) as follows:

1 The coordinator v calculates the preparation data

$$X = g^r \bmod p, \tag{3}$$

where *r* is a random number ($\in Z_q$).

2 The coordinator v decides which entities to revoke. Let Λ be the set of revoked entities and d is the number of the revoked entities.

3 The coordinator v picks k-d-1 integers from a set $\{n+1, ..., n+k$ -1 $\}$ and let Θ be the set of chosen integers. Then the coordinator calculates k-1 revocation data as follows:

$$M_i = y_i^r \bmod p \ (j \in \Lambda \cup \Theta), \tag{4}$$

using the public keys of revoked entities and the spare public keys on the public bullet in board.

4 The coordinator v broadcasts following broadcast data to all entities:

$$B(\Lambda, r) = X \| \{ [j, M_i] | j \in \Lambda \cup \Theta \},$$
 (5)

where || indicates "concatenation" of data.

<Calculation of the group key U by the coordinator>

The coordinator v calculates a group key U using its own secret key s_v and broadcast data B (Λ, r) :

$$U = X^{s_{V} \times L(\Lambda \cup \Theta \cup \{v\}, v)} \times \prod_{j \in \Lambda \cup \Theta} M_{j}^{L(\Lambda \cup \Theta \cup \{v\}, j)} \mod p,$$

$$(6)$$

where

$$L(\Psi, w) = \prod_{t \in \Psi \setminus \{w\}} t/(t-w) \mod q \ (\forall \Psi: \text{set}, \forall w: \text{integer}).$$
 (7)

Since M_j (= $g^{s_j \times r} \mod p$) holds, the system secret key S is recovered on the exponent of equation (6), gathering k sets of secret keys:

$$U = g^{r \times s_v \times L(A \cup \Theta \cup \{v\}, v)}$$

$$\times \prod_{j \in A \cup \Theta} g^{s_j \times r \times L(A \cup \Theta \cup \{v\}, j)} \mod p$$

$$= g^{r \{s_v \times L(A \cup \Theta \cup \{v\}, v) + \sum_{j \in A \cup \Theta} (s_j \times L(A \cup \Theta \cup \{v\}, j))\}} \mod p$$

$$= g^{r \times S} \mod p.$$

Each entity can reuse its secret key s_i , which is a shadow of the system secret key s_i , because the system secret key s_i is recovered on exponent of s_i , not on s_i , not on s_i , which is a shadow of the system secret key s_i , wh

< Calculation of the group key U by a non-revoked entity >

Receiving the broadcast data, a non-revoked entity i calculates the group key U using its own secret key s_i , similar on the coordinator v,

$$U = X^{s_i \times L (\Lambda \cup \Theta \cup \{i\}, i)}$$

$$\times \prod_{j \in \Lambda \cup \Theta} M_j^{L(\Lambda \cup \Theta \cup \{i\}, j)} \mod p$$

$$= g^{r \times S} \mod p.$$
(8)

The system secret key S is recovered on the exponent of equation (8), gathering k secret keys.

On the other hand, a revoked entity j can not calculate the group key U because X^{s_j} which the entity j can calculate by using its own secret key s_j is equal to the revocation data M_j , and the entity j can gather only k-1 secret keys on the exponent.

3.4 Concrete Example

We show the concrete example in Figure.1 for the following one:

- 1 The coordinator (entity 2) decides to revoke the entity 4.
- 2 The coordinator calculates the preparation data $X (=g^r \mod p)$ and the revocation data $M_4(=y_4^r \mod p)$.
- 3 The coordinator broadcasts the broadcast data B $(4, r) = X ||\{[4, M_4]\}\}$.
- 4 The coordinator calculates the group key $U (=g^{r \times S} \mod p)$ by using its own secret key s_2 and the broadcast data B (4, r).
- 5 The Entity 1 calculates the group key U by using its own secret key s_1 and the broadcast data B (4, r).
- 6 The Entity 3 calculates the group key U by using its own secret key s_3 and the broadcast data B (4, r).
- 7 The Entity 4 can't calculate the group key U by using its own secret key s_4 and the broadcast data B (4, r). Because the broadcast data B (4, r) includes the secret key s_4 .

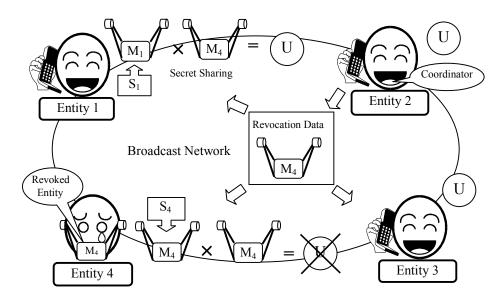


Fig.1. The concrete example of our proposal

4 Security

First, we discuss how difficult is finding the group key U for the revoked entity and an outsider of the group. We will examine three types of attacks to pillage the group key as follows:

- 1 All entities can get $y = g^S \mod p$ by using k sets of public key y_i shown in equation (2). To get the group key U from y, the revoked entity needs to obtain the random number r that the coordinator generates. Since the random number r is an exponent of the preparation data and the revocation data, the level of difficulty in getting r is the same as that of solving DLP.
- 2 To get the group key U from the preparation data X, the revoked entity needs to obtain the system secret key S that the system manager generates. Since the system secret key S is an exponent of the above $y \ (=g^S \mod p)$, the level of difficulty of getting S is also the same as that of solving DLP.
- 3 We assume a trial to get the group key U from the revocation data shown in equation (4). From broadcast data B (Λ, r) , all entities can get k-1 revocation data M_j which includes the secret key s_j respectively on the exponent. Since the system secret key S is divided into the secret keys by threshold k, however, the revoked entity can not calculate the group key U which includes S on the exponent. Even if the revoked entity uses its own secret key s_j , the number of shadows does not increase.

Next, we consider an attack to modify and forge broadcast data. The coordinator generates the preparation data X and the revocation data M_j using only the public information. Therefore, it is necessary to append the coordinator's signature to the broadcast data in order to prevent modification and forgery. In sections 5 and 6, we will explain a method that includes coordinator authentication, in which the amount of broadcast data does not increase, compared with the basic scheme explained in section 3. Moreover, we consider that a time-stamp on the broadcast data is necessary to prevent a replay attack. To prevent an attacker from modifying and forging a public key on a public bulletin board, the board should be managed by a trusted system manager or all public information should be stored with certifications made by a trusted third party.

Finally, we discuss the security of our scheme when entities form a conspiracy. Even if all revoked entities conspire, they can not reconstruct the secret key S since they can get at most d (< k) shadows s_j of S, which is less than the threshold-k. Here, we don't assume a conspiracy attack that a non-revoked entity cooperates with the revoked entity. If the attack is possible, revoked entities can get all group keys and all decrypted messages through the co-conspirator. To prevent this type of attack, different techniques are required, for example traitor tracing or watermark, which is outside of the scope of this paper.

5 Applications

In this section, we describe some considerations that are necessary to apply our scheme to an actual group communication system.

5.1 New Entity

When a new entity wants to join a group communication system, a system manager decides its unique identity number c ($n+k \le c \le q-1$) which is different from ones of the existing entities. The system manager calculates its secret key $s_C = f(c)$ and sends it to the new entity through a secure way. Then, the system manager calculates the public key $y_C = g^{s_C} \mod p$ and adds it to the public bulletin board.

This procedure does not affect the existing entities.

5.2 The Number of Revoked Entities

Our method shown in section 3 enables a coordinator to revoke a maximum of k-1 entities at one time. Also, the parameter k determines the amount of broadcast data from equation (5). If the number of entities that the coordinator can revoke at once becomes large, the broadcast data amount becomes large. Therefore, a system manager should decide the parameter k to fit for an actual system. The coordinator can distribute a group key without "entity revocation", by using k-1 sets of spare public information for the broadcast data.

5.3 Continuity of Revocation

In actual group communication, our scheme is used repeatedly by a different coordinator and revoked entities. A coordinator can decide either one of the following cases:

- -revoke entities that were revoked last time (by indicating the entities as revoked again) or
- -send a new group key to entities which were revoked last time (by not indicating the entities as revoked this time).

Also, the coordinator can use the previous group key to make a new one in order to revoke entities that were revoked last time.

Next, we show a method of revoking specific entities from the group communication completely:

- 1 The system manager distributes a random number *e* to all entities other than the specific ones by our scheme shown in section 3.
- 2 Each entity except the specific ones replaces own secret key s_i with

$$s_i' = s_i \times e \mod q. \tag{9}$$

3 The system manager replaces the system parameter *g* on the pubic bulletin board with

$$g' = g^{1/e} \mod p. \tag{10}$$

With this method, it is not necessary to change every public key on the public bulletin board since $y_i = (g')^{s_i'} \mod p$ is satisfied. It is practical because the public keys might be stored in a local storage by each entity. The revoked entities do not join the group communication permanently because they do not have the secret key s_j' to satisfy $y_j = (g')^{s_j'} \mod p$. When the system manager wants the revoked entity to join the group communication again, the system manager would send its new secret key s_j' through a secure way.

5.4 Some Modifications

Our scheme is considered a one-pass Diffie-Hellman key exchange scheme with "entity revocation". The coordinator distributes the preparation data $X = g^r \mod p$, and shares a group key $U = g^{S \times r} \mod p$ with the other entities, where we regard $Y = g^S \mod p$ as a public key for the group. Thus, a coordinator can select any group key Z by broadcasting $Y = Z \times U \mod p$ together, similar to the ElGamal public key cryptosystem [5].

If an attacker can use a key calculation mechanism of an entity as an oracle, a similar modification using the Cramer-Shoup public key cryptosystem [3] would be effective against an adaptive chosen ciphertext attack.

Also, we can modify our scheme so as to prevent an attacker from modifying and forging broadcast data, by adding a message recovery signature as follows:

<Setup by a system manager>

Same as the basic scheme explained in section 3 except that the system manager publishes a hash function (hash) on a public bulletin board.

<Generation of broadcast data by a coordinator>

- 1 The coordinator calculates revocation data M_j for $j \in \Lambda \cup \Theta$ shown in equation (4), by using a random number r.
- 2 The coordinator calculates a following hash data:

$$H = \text{hash } (v \parallel [j \parallel M_i], \ j \in \Lambda \cup \Theta). \tag{11}$$

3 The coordinator v generates its signature of the hash data, using its secret key s_v :

$$Z = H \times (-s_v) + r \bmod q. \tag{12}$$

4 The coordinator v broadcasts the following broadcast data:

$$B(\Lambda, r) = Z \| v \| \{ [j, M_i] | j \in \Lambda \cup \Theta \}.$$
 (13)

The amount of broadcast data is less than that of our basic scheme explained in section 3 shown in equation (5) since $Z \parallel v$ is less than X (1024bit).

< Key exchanging by a non-revoked entity>

- 1 Similar to equation (11), the entity calculates hash data H'. If the data is not changed, H' = H.
- 2 The entity recovers the preparation data X' using the public key of the coordinator y_v :

$$X' = g^Z \times y_v^{H'} \mod p. \tag{14}$$

If the signature Z originates from the right coordinator $v, X' \equiv X$ shown in equation (3).

The rest of procedure to distribute the group key U is the same as the basic scheme explained in section 3.

This scheme uses one of the message recovery signature schemes proposed in [13]. Therefore, other variations of the signature are possible:

For example,
$$Z' = H \times r + s_v \mod q$$
.

6 Evaluation

In this section, we will evaluate our proposal, comparing with the following four previously reported methods:

- -Familiar Method 1: a method whereby a distributor distributes a group key to *n-d* entities individually, except *d* revoked entities, using a 128bit symmetric key block cipher.
- -Familiar Method 2: the same method as Familiar Method 1, using 1024bit RSA cryptosystems.
- -Previous Method 1: a method in paper [10], using the "RSA common modulus attack".
- -Previous Method 2: another method in paper [10], using the "RSA low exponent attack".

We will evaluate our proposal based on the four requirements that we have already described in sections 1 and 2:

- -Requirement 1: The amount of transmission and the delay do not rely on group scale.
- -Requirement 2: Preparation phase is not necessary.
- -Requirement 3: The fixed-privileged distributor is not required.
- -Requirement 4: The security of the scheme is based on DLP.

First, we will evaluate the performance of our proposal. Figure 2 shows the performance of our proposal, compared with "Familiar Method 1" and " Familiar method 2". In Figure 2, axis x shows the number of entities in the group, and axis y marks the delay (sec) until all entities complete a group key sharing. However, the number of revoked entities is d=1. We assume that the delay is the sum of data transfer time and calculation time for each entity. We estimate data transfer time by assuming the transmission rate to be 28.8kbps. Also, we estimate a calculation time

by using experimental results obtained from a 200MHz Sun Ultrasparc (with gcc 2.7.2.3).

This figure shows the delay of our proposal does not rely on group scale. So, our scheme satisfies requirement 1 shown above. In Figure 2, "Familiar Method 1" seems more efficient than our proposal, because the cross-point of them is rather large (n=180). Though we estimate the data transfer time by the amount of transmission here, the data transfer time is related to the number of communication in actual communication. Some control data is added to the transmitted data for each connection. Also an authentication and a negotiation are necessary for each communication. Therefore, we consider the cross-point of two methods is surely much less than n=180, because the number of communication of "Familiar method" increases, relying on the group scale. On the other hand, the number of communication of our proposal is constant.

Figure 3 shows the performance of our proposal, compared with "Previous Method 1" and "Previous method 2". In Figure 3, axis x shows the number of revoked entities d, and axis y marks the delay (sec) until all entities complete a group key sharing. Here, measurement conditions are the same as in Figure 2. "Previous Method 1" can not revoke two or more entities at once.

We can see that the delay of our proposal is less than that of "Previous Method 2" where $d \ge 45$. Therefore, our proposal can revoke entities quickly, even if a coordinator should revoke many entities at one time. Moreover the calculation time of "Previous Method 2" increases exponentially as d increases, On the other hand, the operation amount of our proposal increases linearly as d increases. The delay of our proposal is within 1 sec in the case of d=1.

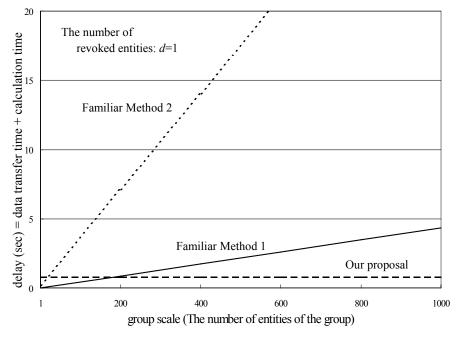


Fig.2. Performance Comparison (with Familiar Methods)

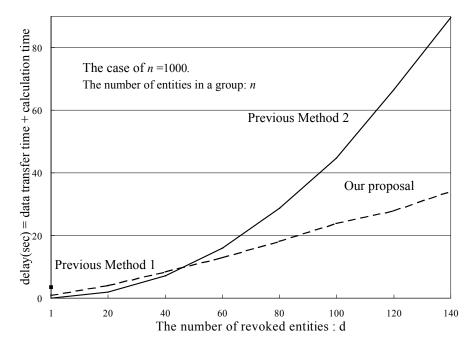


Fig.3. Performance Comparison (with Previous Methods)

Next, we evaluate the features of our proposal. Table 1 shows that our proposal satisfies four requirements, whereas four existing methods do not. Therefore, we consider that our proposal can be applied to many systems with some restrictions.

Requirement	Requirement 1	Requirement 2	Requirement 3	Requirement 4
Method				
Our proposal	yes	yes	yes	yes
Familiar Method 1	no	yes	no	
Familiar Method 2	no	yes	yes	no
Previous Method 1	yes	no	no	no
Previous Method 2	yes	no	no	no

Table1. Comparison of features

7 Conclusions

In this paper we have proposed a quick group key distribution scheme with "entity revocation". The features of our scheme are as follows:

- -The amount of transmission and the delay do not relay on group scale. This feature allows a quick key distribution with "entity revocation" even when the group is large.
- -Preparation phase is not necessary. This feature is suitable for a system with frequent "entity revocation".
- -Any entity can act as coordinator, and revoke any other entities. This feature is suitable for group communication systems in which all members have equal rights like a conference system.
- -Data transfer time and entity calculation time are reduced by using elliptic curve or hyper-elliptic curve cryptosystems.

References

- [1] C.Asmuth, J.Bloom, "A Modular Approach to Key Safeguarding", IEEE Trans. on Information Theory, v. IT-29, n.2, Mar 1983, pp.208-210.
- [2] M.Burmester, Yvo.Desmedt, "A Secure and Efficient Conference Key Distribution System", Advances in Cryptology-EUROCRYPT'94, pp.275-285, Springer-Verlag, 1994.
- [3] R.Cramer, V.Shoup, "A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack", Advances in Cryptology: Proceedings of CRYPTO'98, pp. 13-25, Springer-Verlag, 1986.
- [4] Y.Desmedt, Y.Frankel, "Threshold cryptosystems", Proceedings of Crypto'89, LNCS435, pp.307-315, Springer-Verlag, Aug. 1989.
- [5] T.ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms", IEEE Trans. Inform. Theory, 31: 469-472, 1985.
- [6] J.Hastd, "On using RSA with low exponent in a public key network", Advances in Cryptology: Proceedings of CRYPTO'85, Vol.218, pp. 403-408, Springer-Verlag, 1986.
- [7] M.Hwang, W.Yang, "Conference key distribution schemes for secure digital mobile communications", IEEE Journal on Selected Areas in Communications, vol. 13, No.2, Feb. 1995
- [8] I.Ingemarsson, D.T.Tang, and C.K.Wong, "A conference key distribution system", IEEE Trans. Inform.Theory, vol. IT-28, pp. 714-p. 720, Sep. 1982.
- [9] K.Kurosawa, Y.Desmedt, "Optimum Traitor Tracing and Asymmetric Scheme", Advances in Cryptology-EUROCRYPT'98, pp. 145-157, Springer-Verlag, 1998.
- [10] N.Matsuzaki, J.Anzai, "Secure Group Key Distribution Schemes with Terminal Revocation", Proceedings of 1998 First Japan- Singapore Joint Workshop on Information Security, pp. 37-44, 1998.
- [11] G.J.Simmons, "A 'Weak' privacy protocol using the RSA cryptosystem", Crypto-logia, Vol.7, No.2, pp. 180-182, 1983.
- [12] A.Shamir, "How to share a secret", Comm.Assoc. Comput. Mach., vol. 22, no. 11, pp. 612-3, Nov. 1979.
- [13] K.Nyberg, R.A.Rueppel, "Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem", Proceedings of Eurocrypt'94, LNCS950, pp. 182-193, 1995.

An Efficient Hierarchical Identity-Based Key-Sharing Method Resistant against Collusion-Attacks*

Goichiro Hanaoka¹^{⋆⋆}, Tsuyoshi Nishioka², Yuliang Zheng³ and Hideki Imai¹

The 3rd Department, Institute of Industrial Science, the University of Tokyo 7-22-1 Roppongi, Minato-ku, Tokyo 106-8558, JAPAN Phone & Fax: +81-3-3402-7365

E-Mail:hanaoka@imailab.iis.u-tokyo.ac.jp imai@iis.u-tokyo.ac.jp

² Information Technology R&D Center, Mitsubishi Electric Corporation 5-1-1 Ofuna, Kamakura, 247-8501, JAPAN

Phone: +81-467-41-2181 & Fax: +81-467-41-2185

E-Mail:nishioka@iss.isl.melco.co.jp

The Peninsula School of Computing and Information Technology Monash University, McMahons Road, Frankston Melbourne, VIC 3199, Australia

Email: yzheng@fcit.monash.edu.au

URL: http://www-pscit.fcit.monash.edu.au/~yuliang/
Phone: +61 3 9904 4196, Fax: +61 3 9904 4124

Abstract. Efficient ID-based key sharing schemes are desired worldwidely for secure communications on Internet and other networks. The Key Predistiribution Systems (KPS) are a large class of such key sharing schemes. The remarkable property of KPS is that in order to share the key, a participant should only input its partner's identifier to its secret KPS-algorithm. Although it has a lot of advantages in terms of efficiency, on the other hand it is vulnerable by certain collusion attacks. While conventional KPS establishes communication links between any pair of entities in a communication system, in many practical communication systems such as broadcasting, not all links are required. In this article, we propose a new version of KPS which is called *Hierarchical KPS*. In Hierarchical KPS, simply by removing unnecessary communication links, we can significantly increase the collusion threshold. As an example, for a typical security parameter setting the collusion threshold of the Hierarchical KPS is 16 times higher than that of the conventional KPS while using the same amount of memory at the KPS center. The memory required by the user is even reduced for a factor 1/16 in comparison with the conventional linear scheme. Hence, Hierarchical KPS provides a more efficient method for secure communication.

^{*} A part of this work was performed in part of Research for the Future Program (RFTF) supported by Japan Society for the Promotion of Science (JSPS) under contact no. JSPS-RETF 96P00604.

^{**} A Research Fellow of JSPS

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 348-362, 1999.

[©] Springer-Verlag Berlin Heidelberg 1999

1 Introduction

For information security, ID-based key distribution technologies are quite important. The concept of ID-based key cryptosystems was originally proposed by Shamir[3, 4]. Maurer and Yacobi have presented an ID-based key distribution scheme following Shamir's concept [5, 6]. However, their scheme requires a huge computational power. Okamoto and Tanaka[7] also proposed a key-distribution scheme based on a user's identifier, but it requires prior communications between a sender and a receiver to share the employed key. Although Tsujii and others proposed several ID-based key-distribution schemes[8, 9], almost all of them have been broken[10]. Thus, the performance of these schemes is unsatisfactory. However, Blom's ID-based key-distribution scheme[2], which is generalized by Matsumoto and Imai[1], has quite good properties in terms of computational complexity and non-interactivity. Many useful schemes based on Blom's scheme have been proposed[1, 11, 12, 13, 14, 15, 16, 17], and known as Key Predistribution Systems (KPS).

In a KPS, no previous communication is required and its key-distribution procedure consists of simple calculations. Furthermore in order to share the key, a participant should only input its partner's identifier to its secret KPS-algorithm. Blundo et al.[14, 15, 16], Kurosawa et al.[18] showed a lower bound of memory size of users' KPS-algorithms and developed KPS for a conference-key distribution. Moreover Fiat and Naor[17], Kurosawa et al.[19] applied a KPS for a broadcasting encryption system.

Although KPS has many desired properties, the following problem exists, as well: When a number of users, which exceeds a certain threshold, cooperate they can calculate the central authority's secret information. Setting up a higher collusion threshold in this scheme requires larger amounts of memory in the center as well as for the users. Solution of this problem will make KPS much more attractive for ID-based key-distribution.

Although KPS provides common keys for all possible communication links among entities, in practical communication systems most of them are not necessary. By removing such unnecessary communication links, we can increase the collusion threshold significantly. This will be explained by means of a new version of KPS called *Hierarchical KPS*. Hierarchical KPS demonstrates how to optimize a KPS for a communication system against collusion attacks. Hierarchical KPS is constructed based on the Matsumoto-Imai scheme[1]. Since the key-distribution procedure in the Matsumoto-Imai scheme consists of only simple calculations, computational cost in Hierarchical KPS is also quite small. As an example, for a typical security parameter setting, the collusion threshold of Hierarchical KPS is 16 times higher than that of the conventional KPS while using the same amount of memory in the KPS center. The memory required by the user is even reduced to 1/16 of that for the conventional linear scheme.

Section 2 gives a brief review of the KPS. Afterwards in section 3, Hierarchical KPS is introduced. This is followed by the evaluation and discussion of the security of Hierarchical KPS in section 4. Section 5 closes the paper with some concluding remarks.

2 A brief overview of KPS

A KPS consists of two kinds of entities: One entity is the KPS center, the others are the users who want to share a common key. The KPS center possesses a secret algorithm by which it can generate an individual KPS algorithm for each user. These individual algorithms are (pre-) distributed by the center to their users and allow each user to calculate a common key from the ID of his communication partner. This section explains how the users' secret KPS-algorithms are generated and how users share a common key in the manner of the Matsumoto-Imai scheme. Note that all the calculations in this article are related to the finite field GF(2).

Let the m-dimensional vectors x_A and x_B be the effective IDs of entities A and B, respectively. The $m \times m$ symmetric matrices $G^{(\mu)}$ ($\mu = 1, \dots, h$) are called KPS-center algorithm. The $G^{(\mu)}$ s are produced by the KPS center and kept secret to all other entities. $G^{(\mu)}$ generates the μ -th bit of a communication key between users A and B, so h is the length of this key. $X_A^{(\mu)}$ and $X_B^{(\mu)}$ are the secret KPS-algorithms of A and B, respectively. $X_A^{(\mu)}$ and $X_B^{(\mu)}$ are calculated by the KPS center as follows:

$$X_A^{(\mu)} = x_A \ G^{(\mu)},\tag{1}$$

$$X_B^{(\mu)} = x_B \ G^{(\mu)}. \tag{2}$$

 $X_A^{(\mu)}$ and $X_B^{(\mu)}$ are contained in tamper-resistant-modules (TRM) and distributed to A and B, respectively. (If procedures for inputting data into TRM is thought to be complicated, TRM is not necessary.) By using $X_A^{(\mu)}$ and $X_B^{(\mu)}$, A and B share their symmetric key as follows:

$$A: k_{AB}^{(\mu)} = X_A^{(\mu)} {}^t x_B, \tag{3}$$

$$B: k_{AB}^{(\mu)} = X_B^{(\mu)} {}^t x_A, \tag{4}$$

where $k_{AB}^{(\mu)}$ indicates the μ -th bit of the shared key k_{AB} between A and B. KPS, including the Matsumoto-Imai scheme, has three noteworthy proper-

KPS, including the Matsumoto-Imai scheme, has three noteworthy properties. First, there is no need to send messages for the key distribution between entities who want to establish a cryptographic communication channel. Second, its key-distribution procedure consists of simple calculations so that its computational costs are quite small. Finally, in order to share the key, a participant has only to input its partner's identifier to its secret KPS-algorithm. Thus, KPS is well applicable to one-pass or quick-response transactions, e.g. mail systems, broadcasting systems, electronic toll collection systems, and so on.

However, KPS has a certain collusion threshold; when more users cooperate they can calculate the center-algorithm $G^{(\mu)}$. For example in the Matsumoto-Imai scheme, as already mentioned above $G^{(\mu)}$ is a $m \times m$ matrix. Hence, by using m linearly independent secret KPS-algorithms, the KPS-center algorithm is easily revealed (note however that, in order to participate in this collusion attack, each adversary has to break his TRM). Thus, m is determined by the

•	-		
	consumer	provider	server
consumer	×	0	×
provider	0	Δ	Δ
server	X	Δ	Δ

Table 1. Required communications in practical communication systems, where \bigcirc , \triangle and \times indicate required, partly required and unnecessary, respectively.

number of users. In order to avoid such collusion attacks, we need to increase the value of m. However, since the number of elements of $G^{(\mu)}$ is m^2 , a quite large memory size is required for the KPS center to increase the value of m. Further, the memory size of a user's secret KPS-algorithm is thereby enlarged in proportion to m. Although these memory sizes are not small, they are proven to be optimal[14]. Hence, in the conventional KPS, we cannot cope with collusion attacks efficiently. This can be a serious problem, especially in a situation where the available memory is strictly limited (e.g. IC cards). For example, m=8192 is selected as the collusion threshold in "KPSL1 card"[20], where the key length is 64bits. The secret-algorithm itself then consumes 64-KBytes of memory size in each IC card. Therefore KPS was considered to be somewhat expensive for real IC card systems at that time. Furthermore by introducing $128\sim256$ bits symmetric key cryptosystems, the required memory size will be $128\sim256$ -KBytes.

Although the conventional KPS provides a common key between any pair of entities, most of them are not necessary in practical communication systems. When no keys are provided for such unnecessary communication links, the collusion threshold can be increased and the memory size of the users decreased, while the memory size of the KPS center stays the same.

3 Hierarchical KPS

In practical communication systems, such as broadcasting, entities are classified into 3 classes: consumer, provider, and server. Figure 1 displays the structure of their communication links. Consumers, i.e. the majority of the entities, receive information from any provider. Servers hold information needed by providers. For example, in broadcasting, addressees and broadcasting stations are regarded as consumers and providers, respectively. Certain entities that provide information for broadcasting stations are regarded as servers. In such a communication structure, communication links between consumers are not necessary. Only communication links to providers are required for the consumers. Similarly, although some communication links between providers and servers are required, not all of them are necessary. Furthermore, although communication links among providers/servers are required, not all of them are necessary. So, providers and servers can be divided into multiple groups. Then, we should realize the possibility to share a common key only for

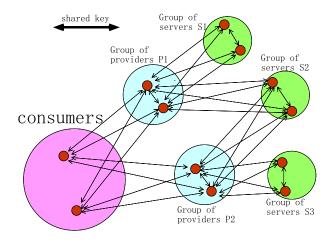


Fig. 1. Communication links in Hierarchical KPS.

- links between consumers and providers,
- links among providers who belong to same group,
- links among servers who belong to same group,
- links between providers and servers, supposed the group of providers and the group of servers are allowed to communicate with each other.

Necessary communication links in this structure are summarized in Table 1.

As mentioned above, in the Matsumoto-Imai scheme the collusion threshold can be increased by replacing the square $m \times m$ matrix $G^{(\mu)}$ of the center algorithm by a larger square matrix; this however requires a significantly larger memory size in the KPS center. Another possibility is to replace the $m \times m$ square matrix by a rectangular $m' \times n'$ matrix of the same size, $m^2 = m' \times n'$, m' > m, n' < m. This requires that the set of users is split into two distinct subsets. The threshold for a collusion of members of the first subset is m', and for a collusion of members of the second subset it is n'. Then a member of one subset can share a common key only with any member of the other subset; common keys between members of the same subset are not possible. This kind of KPS with asymmetric center algorithm will be used below to realize key distribution between consumers and providers, since no common keys are required among consumers.

From the requirement that the memory size of the KPS center should be fixed, i.e. from the equation $m^2 = m' \times n'$, it becomes clear that the collusion threshold m' for the consumers will increase when n' decreases. This means that there should be only few members in the second subset. Therefore a member of this subset is a group of providers who all provide access to several groups of servers. In other words, a "layer" of provider-groups is inserted between the consumers and the groups of servers, see Figure 1. Therefore the new version of

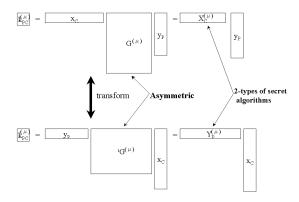


Fig. 2. Key distribution between a consumer and a provider in Hierarchical KPS.

KPS has been called "Hierarchical KPS". The following sections explain it in detail.

3.1 Key distribution between consumers and providers

Our improvement of KPS starts with replacing the symmetric matrices for the KPS center algorithm in the Matsumoto-Imai scheme by asymmetric $m \times n$ matrices $G^{(\mu)}$ ($\mu = 1, \dots, h$). Then key distribution between consumers and providers is implemented in the following way:

Let the m-dimensional vector x_C be the effective ID of consumer C, the n-dimensional vector y_P be the effective ID of provider P. Then C's secret KPS-algorithm $X_C^{(\mu)}$ is calculated by

$$X_C^{(\mu)} = x_C \ G^{(\mu)},\tag{5}$$

and $Y_P^{(\mu)}$ is P's secret KPS-algorithm which is calculated as follows:

$$Y_P^{(\mu)} = y_P \, {}^t G^{(\mu)}. \tag{6}$$

C and P share their symmetric key k_{PC} according to

$$C: k_{PC}^{(\mu)} = x_C G^{(\mu)} {}^t y_P = X_C^{(\mu)} {}^t y_P, \tag{7}$$

$$P: \ k_{PC}^{(\mu)} = y_P^{-t} G^{(\mu)-t} x_C = Y_P^{(\mu)-t} x_C, \tag{8}$$

where $k_{CP}^{(\mu)}$ indicates the μ -th bit of k_{CP} , the shared key between C and P. Figure 2 illustrates the key distribution between a consumer and a provider in Hierarchical KPS.

Key distribution among providers 3.2

In this subsection, we explain key distribution among providers with asymmetric matrices $G^{(\mu)}$.

For key distribution among providers we embed a symmetric matrix $G_{sym}^{(\mu)}$ into $G^{(\mu)}$, where $G^{(\mu)}_{sym}$ consists of rows in $G^{(\mu)}$ (Figure 3). By using $G^{(\mu)}_{sym}$, providers can share their keys as shown in Figure 4. According to the selection of rows belonging to $G_{sym}^{(\mu)}$ in $G^{(\mu)}$, elements from $Y_P^{(\mu)}$ and $Y_{P'}^{(\mu)}$ are selected to form n-dimensional vectors $\overline{Y_P^{(\mu)}}$ and $\overline{Y_{P'}^{(\mu)}}$. Using $\overline{Y_P^{(\mu)}}$ and $\overline{Y_{P'}^{(\mu)}}$, two providers P and P' share their key as follows:

$$P: k_{PP'}^{(\mu)} = \overline{Y_P^{(\mu)}} \, {}^t y_{P'},$$
 (9)

$$P': k_{PP'}^{(\mu)} = \overline{Y_{P'}^{(\mu)}} \, {}^{t}y_{P}, \tag{10}$$

Again, $k_{PP'}^{(\mu)}$ indicates the μ -th bit of the shared key $k_{PP'}$ between P and P'. Although providers can share their keys by using this method, there are the

following problems:

- $-G_{sym}^{(\mu)}$ might be revealed by consumers' collusion attacks, if the selection of rows belonging to $G_{sym}^{(\mu)}$ in $G^{(\mu)}$ is exposed (for convenience, call this selection
- A key between two providers cannot be longer than a key between a provider and a consumer.

As already mentioned, we assume that there are some groups of providers and that a provider communicates only with other providers in his group. The above problem can be solved if more than one $G_{sym}^{(\mu)}$ can be extracted from one $G^{(\mu)}$ and more than one $k_{sel}^{(\mu)}$ is distributed in each group of providers.

Suppose that $G_{sym}^{(\mu),ij}$ $(i=1,\cdots,N_{sym},\ j=1,\cdots,N_P)$ are $n\times n$ symmetric matrices embedded in $G^{(\mu)}$, and $k_{sel}^{(\mu),ij}$ $(i=1,\cdots,N_{sym},\ j=1,\cdots,N_P)$ are the

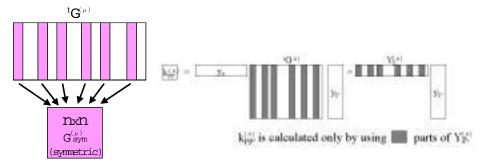


Fig. 3. Embedded symmetric matrix $G_{sym}^{(\mu)}$ in $G^{(\mu)}$.

Fig. 4. Key distribution among providers.

selection of rows belonging to $G_{sym}^{(\mu),ij}$ in $G^{(\mu)}$. N_{sym} is the number of embedded symmetric matrices that are distributed within one group of providers, and N_P is the number of groups of providers. Note that $N_{sym}N_P$ should be no more than m/n for the security of the system. $k_{sel}^{(\mu),ij}$ $(i=1,\cdots,N_{sym})$ are distributed to all providers in the j-th group P_j . Then, key distribution between providers P and P', who both belong to P_j , is carried out as follows:

$$P: k_{PP'}^{(\mu),ij} = \overline{Y_P^{(\mu),ij}} \, {}^{t}y_{P'} \, (i = 1, \cdots, N_{sym}), \tag{11}$$

$$P': k_{PP'}^{(\mu),ij} = \overline{Y_{P'}^{(\mu),ij}} {}^{t}y_{P} \quad (i = 1, \cdots, N_{sym}), \tag{12}$$

where $k_{PP'}^{(\mu),ij}$ $(i=1,\cdots,N_{sym})$ indicates the μ -th bit of the shared key $k_{PP'}^{ij}$ $(i=1,\cdots,N_{sym})$ between P and P', and elements from Y_P and $Y_{P'}$ are selected according to $k_{sel}^{(\mu),ij}$ $(i=1,\cdots,N_{sym})$ to form n-dimensional vectors $\overline{Y_P^{(\mu),ij}}$ and $\overline{Y_{P'}^{(\mu),ij}}$ $(i=1,\cdots,N_{sym})$.

So, if a $G_{sym}^{(\mu),i_0j}$ has been exposed by a certain consumers' attack, the providers in P_j can deal with this attack by using another $k_{sel}^{(\mu),i_1j}$, $i_1 \neq i_0$. Furthermore, by using multiple $k_{sel}^{(\mu)}$ simultaneously, providers can share longer keys. For example, if both $k_{sel}^{(\mu),i_0j}$ and $k_{sel}^{(\mu),i_1j}$ are used simultaneously, the length of the keys among providers in P_j can be 2h, that is twice the length of the keys between consumers and providers. Accordingly, the keys shared among providers can be at most $N_{sym}h$.

Additionally, note that this scheme permits a provider to belong to multiple groups concurrently.

3.3 Key distribution between providers and servers

As already mentioned, servers can share keys with providers, assuming that the groups they belong to are allowed to communicate with each other. In this subsection, we show how to produce a server's secret KPS-algorithm.

Let the *n*-dimensional vectors z_S be the effective ID of server S and $Z_S^{(\mu),ij}$ be the secret KPS-algorithm of S which is calculated as follows:

$$Z_S^{(\mu),ij} = z_S \ G_{sym}^{(\mu),ij} \ (i = 1, \cdots, N_{sym}),$$
 (13)

Herein it is assumed that S belongs to group of servers S_j that is allowed to communicate with the providers in group P_j . By using this secret KPS- algorithm, communication keys are shared between S and P as follows:

$$S: k_{SP}^{(\mu),ij} = Z_S^{(\mu),ij} {}^{t}y_P \quad (i = 1, \cdots, N_{sym}), \tag{14}$$

$$P: \ k_{SP}^{(\mu),ij} = \overline{Y_P^{(\mu),ij}} \ ^tz_S \ (i = 1, \cdots, N_{sym}), \tag{15}$$

where $k_{SP}^{(\mu),ij}$ $(i=1,\cdots,N_{sym})$ indicates the μ -th bit of the shared key k_{SP}^{ij} $(i=1,\cdots,N_{sym})$ between S and P.

Similarly to the key distribution among providers, if $G_{sym}^{(\mu),i_0j}$ is exposed by a certain attack, S and P can still share their key using other $Z_S^{(\mu),i_1j}$ and $\overline{Y_P^{(\mu),i_1j}}$, $i_1 \neq i_0$. And by concurrent use of their secret KPS-algorithms again longer keys can be used. For example, if $Z_S^{(\mu),i_0j}$, $Z_S^{(\mu),i_1j}$ and $\overline{Y_P^{(\mu),i_0j}}$, $\overline{Y_P^{(\mu),i_1j}}$ are used, the length of a shared key is 2h. As above, the length of the key shared between providers and servers in this manner can be $N_{sym}h$ as the maximum.

Note that a group of servers can be allowed to communicate with multiple groups of providers in this way, and that a server can belong to multiple groups of servers.

3.4 Key distribution among servers

Any pair of servers in the same group can share their communication key using the servers' secret KPS-algorithms mentioned in 3.3. Namely, a pair of servers S and S', who belong to S_i , share their common key as follows:

$$S: k_{SS'}^{(\mu),ij} = Z_S^{(\mu),ij} {}^{t} z_{S'} \quad (i = 1, \dots, N_{sym}), \tag{16}$$

$$S': k_{SS'}^{(\mu),ij} = Z_{S'}^{(\mu),ij} {}^{t}z_{S} \quad (i = 1, \dots, N_{sym}), \tag{17}$$

where $k_{SS'}^{(\mu),ij}$ $(i=1,\cdots,N_{sym})$ indicates the μ -th bit of the shared key $k_{SS'}^{ij}$ $(i=1,\cdots,N_{sym})$ between S and S', $z_{S'}$ is the effective ID of S', and $Z_{S'}^{(\mu),ij}$ $(i=1,\cdots,N_{sym})$ are the secret KPS-algorithms of S' that are produced similarly to those of S.

Similarly to the key distribution among providers or that between providers and servers, if $G_{sym}^{(\mu),i_0j}$ is exposed by a certain attack, S and S' can still share their key using other $Z_S^{(\mu),i_1j}$ and $Z_{S'}^{(\mu),i_1j}$. And concurrent use of their secret KPS-algorithms again results in longer keys. Using $Z_S^{(\mu),i_0j}$, $Z_S^{(\mu),i_1j}$ and $Z_{S'}^{(\mu),i_0j}$, $Z_{S'}^{(\mu),i_1j}$, the length of the shared key is doubled. By this, the keys shared among providers can be at most $N_{sym}h$ long.

4 Evaluation and security discussion

4.1 Communications with Hierarchical KPS

Here we confirm whether Hierarchical KPS can provides the required communications-links in practical communication systems or not. As already discussed, required communication-links are consumer-provider, provider-provider (within a group of providers), provider-server (if the group that the provider belongs to and the group that the server belongs to are allowed to communicate with each other), server-server (with in a group of servers). It can be seen that these communications are available by the method described in $3.1 \sim 3.4$. Hence, it is confirmed that all required functions are provided by Hierarchical KPS.

Furthermore, Hierarchical KPS offers a higher level of security than the Matsumoto-Imai scheme. As mentioned in $3.2 \sim 3.4$, the keys among providers,

colluders	$G^{(\mu)}$	$G_{sym}^{(\mu),ij}$
providers	n	n
consumers	m	$m - n + \log_2 n$
servers	_	$n\dagger$
providers + servers	n providers	n^{\ddagger}

Table 2. Collusion thresholds to calculate $G^{(\mu)}$, $G_{sym}^{(\mu),ij}$.

Table 3. Required memory size for each type of entities.

	KPS center	consumer	1	
Hierarchical KPS	hnm	hn	hm	$hN_{sym}n$
conventional KPS	hnm	$h\sqrt{nm}$	$h\sqrt{nm}$	$h\sqrt{nm}$

those between providers and servers and those among servers can be $N_{sym}h$ bits long, what is more than the length h of keys between consumers and providers. Hence, these communications can be carried out more safely than those by the Matsumoto-Imai scheme, assuming that the number h of matrices for the KPS-center algorithm is the same in Hierarchical KPS and the Matsumoto-Imai scheme.

4.2 Collusion attack against $G^{(\mu)}$

There are mainly three kinds of collusion attacks against $G^{(\mu)}$: the consumers' collusion, the providers' collusion, and the mixed collusion of consumers and servers. The servers cannot reveal $G^{(\mu)}$ by themselves. Although servers and consumers can collude to reveal $G^{(\mu)}$, the influence of the servers in the attack is quite limited. Hence, attacks of the servers against $G^{(\mu)}$ are not regarded as a problem.

To break the whole system, a collusion of m consumers or n providers is needed from the view of information theory because the quantity of the center's secret information is hmn bits, while consumer's secret KPS-algorithm has hn bits information and provider's secret KPS-algorithm has hm bits information.

It should be noted that the mixed collusion between consumers and providers is inefficient since the informations available to consumers and to providers are not independent from each other. The number of either consumers or providers joining in the collusion attack must exceed the corresponding threshold m or n to succeed in the attack.

Actually, since in $G^{(\mu)}$ symmetric matrices are embedded, leakage of one $k_{sel}^{(\mu),ij}$ brings $\frac{n-1}{2}$ reduction of the collusion threshold of consumers. However,

[†] Collusion by servers that belong to group of servers S_j . ‡ Collusion by any providers and severs that belong to S_j .

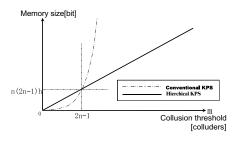


Fig. 5. Comparison of the required memory size for the KPS-center algorithm in Hierarchical KPS with that in conventional KPS, where n indicates the collusion threshold for providers.

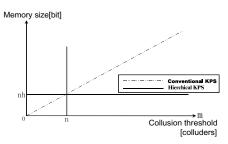


Fig. 6. Comparison of the required memory size for a consumer's secret KPS-algorithm in Hierarchical KPS with that in conventional KPS, where n indicates the collusion threshold for providers.

although all $k_{sel}^{(\mu),ij}$ s are exposed, the collusion threshold is still high enough because $m\gg n$ (note that the collusion threshold of providers cannot be reduced). In Table 2, collusion thresholds against $G^{(\mu)}$ are shown. This means that a Hierarchical KPS can be designed as shown above based on the collusion thresholds n for consumers and m for providers. In conventional KPS, however mixed collusions can also be effective. This is why in conventional KPS the collusion threshold should be (n+m), so that as center algorithm $(n+m)\times (n+m)$ matrices are needed in the Matsumoto-Imai scheme. Based on this assumption, the memory requirements of Hierarchical KPS and conventional KPS will be compared in the next section.

4.3 Memory requirements

Considering these collusion thresholds, m and n are determined mainly by the numbers of consumers and providers, respectively. Similarly, the required memory size for the KPS-center algorithm is determined to be proportional to n times m, while in the Matsumoto-Imai scheme the required memory size for the KPScenter algorithm is determined to be proportional to $(n+m)^2$. Furthermore, the memory size for the consumers' secret KPS-algorithms is proportional to n. Since in the Matsumoto-Imai scheme this is proportional to (n+m), the memory size for the consumers' secret KPS-algorithms can be reduced considerably. Note that for general purpose applications the Matsumoto-Imai scheme achieves the optimal memory size for both the KPS-center and users like Blundo's scheme [14] and some others. Thus, we regard the memory size in the Matsumoto-Imai scheme as that of conventional KPS. As the number of consumers will usually be much higher than the number of providers, these reductions of memory size are quite significant. Figure 5 and Figure 6 show the memory size required for the KPScenter and a consumer. In the Matsumoto-Imai scheme, the required memory size for the KPS-center algorithm grows proportionally to the square of the col-

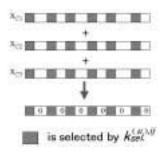


Fig. 7. The required combination of consumers' ID to reveal $G_{sym}^{(\mu),ij}$, where x_{C1} , x_{C2} and x_{C3} are effective identifiers of consumer C1, C2 and C3, respectively.

lusion threshold and the required memory size for users' secret KPS-algorithms proportionally to the collusion threshold. In contrast, in Hierarchical KPS the required memory size for the KPS-center algorithm increases proportionally to the collusion threshold for consumers, and the required memory size for consumers' secret KPS-algorithms remains unchanged while increasing the collusion threshold for consumers, assuming that the collusion threshold for providers is fixed (since the number of providers is much smaller than that of consumers, a quite low collusion threshold is sufficient to avoid a providers' collusion attack). Additionally, since the KPS-center algorithm of Hierarchical KPS consists of $N_{sym} \cdot N_P$ symmetric $n \times n$ matrices, we can reduce the required memory size for the KPS-center algorithm using their symmetrical property; in this way, the difference between the required memory size for the KPS-center algorithm in Hierarchical KPS and that for conventional KPS can be even more remarkable.

Also when taking into account the higher collusion threshold, the difference of the required memory size between Hierarchical KPS and conventional KPS is even more significant.

In summary, the collusion threshold in Hierarchical KPS can be much higher than that of conventional KPS by using same size of memory in the KPS center. Table 3 shows the required memory sizes for each type of entity. Only the memory size for the providers is larger in Hierarchical KPS than in the Matsumoto-Imai scheme. But this is not a serious problem since for providers such an amount of memory should be easily available.

4.4 Collusion attack against $G_{sym}^{(\mu),ij}$

Here, we especially discuss the collusion attack of consumers against $G_{sym}^{(\mu),ij}$ in more detail.

Note that in order to reveal $G_{sym}^{(\mu),ij}$, the adversary requires n combinations of the consumers' secret KPS-algorithms that fulfill the following condition:

Condition (*) For the linear sum of the consumers' IDs participating in the

combination, all the elements except those selected by $k_{sel}^{(\mu),ij}$ are entirely 0 (see Figure 7).

By using n such combinations, $G_{sym}^{(\mu),ij}$ can be revealed easily, when the involved sums are linearly independent. Hence, this attack can be realized by only n colluders in the worst case. However, the possibility of its success seems infeasible. Here, we estimate the number of colluders that yields feasible possibility to realize the attack.

When t consumers collude, the number of combinations of consumers' IDs is $2^t - 1$. Since the probability that a randomly selected combination fulfills the condition (*) is 2^{n-m} , the expectation $E_{col}(t)$ of the number of the combinations that fulfill the condition (*) is approximated as follows:

$$E_{col}(t) = (2^t - 1)(2^{n-m}) \simeq 2^{t+n-m}.$$
 (18)

Thus, to achieve $E_{col}(t) \geq n$, we require $t \geq m-n+\log_2 n$. Hence, $m-n+\log_2 n$ can be regarded as the collusion threshold of this attack. Although this threshold seems to be still high enough, we can find that less than n colluders are required to reveal $G_{sym}^{(\mu),ij}$ if $k_{sel}^{(\mu),ij}$ is exposed. Thus, $k_{sel}^{(\mu),ij}$ must be kept secret to other entities besides its legal users. Basically, m and n were defined according to the number of consumers and providers, respectively. However, since the collusion threshold to reveal $G_{sym}^{(\mu),ij}$ by consumers is defined by both m and n, this must also be considered when choosing m and n. In The collusion thresholds against $G_{sym}^{(\mu),ij}$ is summarized Table 2. Although $k_{sel}^{(\mu),ij}$ can be revealed without difficulty if $G^{(\mu)}$ is exposed, we don't need to take care of this attack since the collusion threshold of $G^{(\mu)}$ is set up high enough to prevent any possible collusion attacks in real world.

As mentioned in **3.2**, by embedding multiple symmetric matrices in $G^{(\mu)}$, the damage of exposing $k_{sel}^{(\mu),ij}$ can be reduced. Namely, if a $G_{sym}^{(\mu),ij}$ is revealed, only the group that uses this $G_{sym}^{(\mu),ij}$ is affected. Although $G_{sym}^{(\mu),ij}$ is damaged, the communication can be realized by using another $G_{sym}^{(\mu),ij}$.

Additionally, although a collusion attack of providers can also reveal $G^{(\mu)}$, the collusion threshold of this attack is the same as that against $G^{(\mu),ij}_{sym}$ by the providers. Hence, in order to reveal $G^{(\mu),ij}_{sym}$, the providers have to reveal $G^{(\mu)}$. Besides, by a collusion attack of servers, $G^{(\mu),ij}_{sym}$ can be revealed. However, only the servers that belong to S_j can carry out this attack. In this attack, the collusion threshold is n, and it is regarded as high enough because there are not so many servers in comparison to consumers (although any provider can also participate in this collusion attack, this attack is still not serious).

4.5 Applications

Hierarchical KPS can be applied to quite many kinds of communication systems. In practical communication systems, we often find two kinds of entities that are regarded as consumers and providers. Usually, a minority of entities in

the system communicates with almost all of the other entities, while the majority communicates only with specific entities (the minority). Hence, we can regard the minority and the majority as providers and consumers, respectively. Furthermore, in communication systems, we also often find entities that provide information to specific providers. Such entities are regarded as servers.

As an example, in broadcasting, addressees and broadcasting stations can be regarded as consumers and providers, respectively. Certain entities that serve information for the broadcasting stations take the role of servers. Assuming that the numbers of addressees, broadcasting stations and servers are 10,000,000, 5,000 and 200,000, respectively, we can set up m=131,072 and n=512 approximately. Then the number $N_{sym}\cdot N_P$ of embedded symmetric matrices is 256. Thus, the collusion threshold of addressees is m=131,072, which is 16 times as large as the 8,192 with the conventional KPS, assuming that the utilized memory size is same in both Hierarchical KPS and the Matsumoto-Imai scheme. In this case, for the Matsumoto-Imai scheme 8192 \times 8192 symmetric matrices are used as the KPS-center algorithm. Even when all the information for the location of embedded symmetric matrices in the center algorithm is exposed, the collusion threshold is still 8 times that of conventional KPS. Furthermore, Memory requirement (using h=64bits) is hn=32,768bits(=4-KBytes), which is 1/16 the requirement of 64-KBytes of the conventional KPS.

5 Conclusion

In this paper, Hierarchical KPS, which is a new style of KPS, has been proposed. It has been pointed out that certain collusion attacks can be effective against KPS, and on the other hand, it has been shown how KPS can be improved for practical communication systems to increase its resistance against collusion attacks. To be specific, by removing communication links that are not required in a practical communication system, resistance against collusion attacks is increased significantly. For a typical security parameter setting, the collusion threshold of the improved KPS is 16 times higher than that of the conventional KPS while using the same amount of memory in the KPS center. The memory required by the users is even reduced to be 1/16 of that for the conventional KPS. Hence, Hierarchical KPS provides a higher level of security against collusion attacks and a simplified implementation due to its reduced memory sizes. This makes Hierarchical KPS attractive for various applications like broadcasting or E-commerce in the Internet. Additionally, since public-key cryptosystems do not have advantages of KPS in terms of computational cost, ID-basedness, and so on, the efficient combination of a public-key cryptosystem and our scheme will realize a more efficient and secure communication system than one single use of a publickey cryptosystem.

References

 T. Matsumoto and H. Imai, "On the KEY PREDISTRIBUTION SYSTEM: A Practical Solution to the Key Distribution Problem," Proc. of CRYPTO'87, LNCS

- 293, Springer-Verlag, pp.185-193, 1987.
- R. Blom, "Non-public Key Distribution," Proc. of CRYPTO'82, Plenum Press, pp.231-236, 1983.
- A. Fiat and A. Shamir, "How to Prove Yourself: Practical Solutions to Identification and Signature Problems," Proc. of CRYPTO'86, LNCS 263, Springer-Verlag, pp.186-194, 1986
- 4. A. Shamir, "Identity-Based Cryptosystems and Signature Schemes," Proc. of CRYPTO'84, LNCS 196, Springer-Verlag, pp.47-53, 1985.
- U. Maurer and Y. Yacobi, "Non-interactive Public-Key Cryptography," Proc. of Eurocrypt'91, LNCS 547, Springer-Verlag, pp.498-407, 1992.
- U. Maurer and Y. Yacobi, "A Remark on a Non-interactive Public-Key Distribution System," Proc. of Eurocrypt'92, LNCS 658, Springer-Verlag, pp.458-460, 1993.
- E. Okamoto and K. Tanaka, "Identity-Based Information Security management System for Personal Computer Networks," IEEE J. on Selected Areas in Commun., 7, 2, pp.290-294, 1989.
- 8. H. Tanaka, "A Realization Scheme of the Identity-Based Cryptosystems," Proc. of CRYPTO'87, LNCS 293, Springer-Verlag, pp.340-349, 1988.
- S. Tsujii and J. Chao, "A New ID-Based Key Sharing System," Proc. of CRYPTO'91, LNCS 576, Springer-Verlag, pp.288-299, 1992.
- D. Coppersmith, "Attack on the Cryptographica Scheme NIKS-TAS," Proc. of CRYPTO'94, LNCS 839, Springer-Varlag, pp.40-49, 1994.
- L. Gong and D. J. Wheeler, "A Matrix Key-Distribution Scheme," Journal of Cryptology, vol. 2, pp.51-59, Springer-Verlag, 1993.
- W. A. Jackson, K. M. Martin, and C. M. O'Keefe, "Multisecret Threshold Schemes," Proc. of CRYPTO'93, LNCS 773, pp.126-135, Springer-Verlag, 1994.
- Y. Desmedt and V. Viswanathan, "Unconditionally Secure Dynamic Conference Key Distribution," IEEE, ISIT'98, 1998.
- C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, "Perfectly Secure Key Distribution for Dynamic Conferences," Proc. of CRYPTO '92, LNCS 740, Springer-Verlag, pp.471-486, 1993.
- C. Blundo, L.A. Frota Mattos and D.R. Stinson, "Trade-offs between Communication and Strage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution," Proc. of CRYPTO '96, LNCS 1109, Springer-Verlag, pp.387-400, 1996.
- C. Blundo and A. Cresti, "Space Requirements for Broadcast Encryption," Proc. of Eurocrypt '94, LNCS 950, Springer-Verlag, pp.287-298, 1995.
- A. Fiat and M. Naor, "Broadcast Encryption," Proc. of CRYPTO '93, LNCS 773, Springer-Verlag, pp.480-491, 1994.
- K. Kurosawa, K. Okada and H. Saido, "New Combinatorial Bounds for Authentication Codes and Key Predistribution Schemes," Designs, Codes and Cryptography, 15, pp.87-100, 1998.
- K. Kurosawa, T. Yoshida, Y. Desmedt and M. Burmester, "Some Bounds and a Construction for Secure Broadcast Encryption," Proc. of ASIACRYPT '98, LNCS 1514, Springer-Verlag, pp.420-433, 1998.
- T. Matsumoto, Y. Takashima, H. Imai, M. Sasaki, H. Yoshikawa, and S. Watanabe, "A Prototype KPS and Its Application IC Card Based Key Sharing and Cryptographic Communication -," Trans. of IEICE Vol. E 73, No. 7, July 1990, pp.1111-1119, 1990.

Periodical Multi-Secret Threshold Cryptosystems

Masayuki Numao

Tokyo Research Laboratory, IBM Japan, Ltd. 1623-14, Shimo-Tsuruma, Yamato, Kanagawa 242-8502, JAPAN numao@jp.ibm.com

Abstract. A periodical multi-secret threshold cryptosystem enables a sender to encrypt a message by using a cyclical sequence of keys which are shared by n parties and periodically updated. The same keys appear in the same order in each cycle, and thus any subset of t+1 parties can decrypt the message only in the periodical time-frames, while no subset of t corrupted parties can control the system (in particular, none can learn the decryption key). This scheme can be applied to a timed-release cryptosystem whose release time is determined when the number of share update phases equals the period of the sequence. The system is implemented by sharing a pseudo-random sequence generator function. It realizes $n \geq 3t+1$ robustness, and is therefore secure against an adversary who can corrupt at most one third of the parties.

1 Introduction

The concept of "timed-release crypto" was first introduced by May [May93] and further studied by Rivest et al. [RSW96]. Its goal is to encrypt a message so that it cannot be decrypted by anyone, not even the sender, until a predetermined amount of time has passed. According to [RSW96], there are two known approaches for implementing it:

- 1. Use trusted agents who promise not to reveal certain information until a specified time.
- 2. Use "time-lock puzzles"-computational problems that cannot be solved without running a computer for at least a certain amount of time.

The problem with the first approach is that the user has to totally trust the agents in all matters from the maintenance of the key to the provision of the decryption service at the specified time. Moreover, since there is no direct correspondence between the key and the time, the handling of the decryption time by the agents is purely operational. The second approach is even less practical, because the decryption time is calculated from the amount of computational steps. Therefore, the receiver has to start the decryption process as soon as he receives the encrypted message, and has to use the best computer available, since the decryption might otherwise be delayed.

In this paper, we will propose an alternative method for realizing "timed-release crypto," using a threshold cryptosystem [Des88] [DF90] that shares a function (possibly a pseudo-random sequence generator) for generating a sequence of random values with some period. Note that in any pseudo-random number generator, the sequence is repeated after a specific period. Our idea is to use the period of the sequence as the time needed to obtain a pair of encryption and decryption keys, so that if the user encrypts the message by using the public key, the corresponding secret key is available only in a periodical time-frame. Since the key pairs are periodically available in the same timeframe, this scheme does not satisfy the original definition of a timed-release key. But it does if the sender encrypts the message by using the encryption key in some time-frame and sends the message in the next timeframe. In this case, the receiver has to wait until one cycle of the period passes to get the decryption key.

Our approach bears some similarity to proactive secret sharing [OY91] [HJKY95] [FGMY98], in the sense that each party updates its share by multiparty computation in each time frame. But the difference is that the proactive scheme aims at maintaining one secret for a long period, whereas in our scheme the secrets themselves are updated so that different secrets appear in different timeframes.

Although our scheme maintains multiple secrets, it differs from the (c,d;k,n)-multi-secret sharing scheme [FY92], in which k different secrets are maintained among n parties in such a way that at least d parties are necessary to recovery all k secrets, whereas no subset of c parties can deduce anything. In our system, the secret (seed) is maintained in a (t,n)-threshold scheme, but it is updated by a function-sharing scheme that results in the generation of a sequence of secrets.

Our approach is also different from the proactive random number generator [CH94], because it deals only with the generation of fresh random values that are computationally independent of the previous states, but cannot maintain the period of the sequence, which is essential to timed-release key generation.

1.1 Our Results

We define a new class of threshold cryptosystems that can be used for timedrelease crypto, and describe an efficient and robust implementation. More precisely, in this paper:

- 1. We define periodical multi-secret threshold cryptosystems as a class of threshold cryptosystems.
- 2. We implement a periodical multi-secret threshold cryptosystem by sharing a pseudo-random sequence generation (PRSG) function.
- 3. We show that our implementations have t-resiliency in $n \geq 3t + 1$, where n and t are the total number of parties and the number of parties corrupted by a mobile adversary, respectively.

1.2 Organization of the Paper

The paper is organized as follows. In section 2, we define our periodical multi-secret threshold cryptosystem. Since we have to share a pseudo-random sequence generator function, we give a multiparty protocol for a sequence generator in section 3. Then, in section 4, we describe periodical multi-secret threshold cryptosystems that we implemented by sharing a linear congruence generator (LCG) and a Blum-Blum-Shub (BBS) generator based on the ElGamal encryption scheme. Finally, we present our conclusions in section 5. In appendix A, we explain some of the cryptographic techniques that are used in our protocols as basic tools.

2 Model and Definition

2.1 Periodical Multi-secret Threshold Cryptosystem

Let E be a public key scheme defined by three protocols: key-pair generation, encryption, and decryption. Key update is then introduced as an additional function of key-pair generation to update the key-pair from (EK_t, DK_t) to (EK_{t+1}, DK_{t+1}) .

A periodical multi-secret threshold cryptosystem $T\bigcup_{Tp_E}$ for scheme E distributes the operation of key generation (update), and decryption among a set of n parties P_1, \ldots, P_n . Let DK_t and $DK_{i,t}$ be a secret key and its share for party P_i in timeframe t respectively; then (EK_t, DK_t) forms a key-pair in timeframe t. $T\bigcup_{T_{nE}}$ is defined by two protocols:

- $T \bigcup_{T_p}$ -KeyUpdate, a randomized protocol that takes a previous share $DK_{i,t-1}$ as private input for party P_i , and returns as public output the public encryption key EK_t for timeframe t and as private output for party P_i a value $DK_{i,t}$, such that (1) $DK_{1,t}, \ldots, DK_{n,t}$ constitute a k-out-of-n threshold secret sharing of DK_t , which is a secret key for timeframe t corresponding to the public key EK_t , and (2) $DK_t = DK_{t'}$ only when $t' = t \pmod{T_p}$, where T_p is the period specified by the system.
- $T \bigcup_{T_p}$ -**Decrypt**, a protocol in which each party P_i takes as public input a ciphertext $C = E_{EK_t}(M)$ and as secret input his share $DK_{i,t'}$ and returns as public output the message M only when $t' = t \pmod{T_p}$.

2.2 Communication Model

We assume that all the communication links are secure (i.e., private and authenticated), which allows us to focus on high-level aspects of the protocols.

2.3 Adversary

We assume that an adversary, A, can corrupt up to t out of the n parties in the network. Since the system changes its internal states by updating the parties'

shares, the ability to deal with only a static adversary is not sufficient to provide security. A "mobile malicious adversary" is allowed to move among parties over time with the limitation that it can only control up to t parties in a timeframe. Here we assume that the adversary is static within a timeframe and moves to other parties in the next timeframe.

2.4 Resiliency

The resiliency of a distributed protocol is defined by comparing it with its corresponding centralized protocol: t-resiliency means that the protocol will compute a correct output even in the presence of a malicious adversary who can corrupt up to t parties (robustness), and that the adversary cannot obtain any information other than what he can obtain from the centralized protocol (privacy).

3 Multiparty Computation for Sequence Generation

In this section, we will describe a multiparty protocol for the sequence generation, since we want to share a pseudo-random sequence generation function to update the secret keys. As sequence generators, we will use the linear congruence generator and the BBS generator (see appendix A.1), because they are simple (need only one multiplication per next number generation) and the periods can be determined by the parameters (the coefficients, modulo, and seed).

On the basis of the multiparty protocol [BGW88] [CCD88], every computation can be distributed in a secure and robust way. The authors of the above papers proved that the bound of the privacy threshold is k < n/2, where n is the total number of parties and k is the number of parties colluding to obtain the secret. They also proved that the robustness threshold is k' < n/3, where k' is the number of parties corrupted by a malicious adversary. Since the above papers assume a computationally unbounded adversary, the share verification process is carried out by error-correcting codes.

By accepting the use of encryption (which satisfies only weaker notions of security), we can employ a more practical non-interactive verification scheme using homomorphic commitment. In this paper, we will use a verifiable secret-sharing scheme (VSS) proposed by Feldman [Fel87] as a basic tool (see appendix A.3). Let $a, b \in Z_p$ be two secrets that are shared by using polynomials $f_a(x), f_b(x)$ of order k, respectively, and let $c \in Z_q, c \neq 0$ be some constant. The values a+b and $c \cdot a$ can be simply computed by having each party perform the same operation on its shares $f_a(i) + f_b(i)$ and $c \cdot f_a(i)$, respectively; this results in the sharing of new polynomials $f_{a+b}(x) = f_a(x) + f_b(x)$ and $f_{ca}(x) = c \cdot f_a(x)$, whose free coefficients are a+b and $c \cdot a$, respectively. Feldman's VSS is also easy to perform, because the correctness of the resulting shares is checked by using the publicly known values (commitments) computed as follows: $g^{a_i+b_i} = g^{a_i}g^{b_i}$ and $g^{c \cdot a_i} = (g^{a_i})^c$.

3.1 Degree Reduction

To share the multiplication $a \cdot b$, three computing steps are necessary: (1) share multiplication, (2) randomization, and (3) degree reduction, because step (1) generates a new polynomial $h(x) = f_a(x)f_b(x)$, which is of order 2k and not irreducible [BGW88].

Here, we show a VSS-based robust degree reduction protocol. Let f(x) be a polynomial of order k to share the secret s. The share $s_i = f(i)$ was distributed by a dealer to party i by means of Feldman's VSS. Now, the parties want to reduce the threshold from k to k' (k' < k) (without relying on the dealer, of course). First, each party i generates a random polynomial $f^{(i)}(x)$ of order k' whose free coefficient is set to s_i , and shares the polynomial by using Feldman's VSS. Then, other parties verify their shares sent from party i and announce whether they are correct or not. If party i has more than k correct announcements, it is qualified. Finally, after k+1 qualified parties are determined, each party j computes his new share using the Lagrange interpolation $s'_j = \sum_{i \in A} \lambda_i f^{(i)}(j)$, where Λ_i denotes a set of qualified parties and λ_i denotes the coefficient computed by the Lagrange interpolation. Note that the coefficients are constant when the set is determined. The protocol is given below:

3.2 Protocol for Robust Degree Reduction

Input of Party i: f(i) of a polynomial f(x) of order k representing a secret s.

Public Input: $\alpha_i = g^{f_i}$ for $0 \le i \le k$, where f_i is the *i*-th coefficient of the f(x). Note that $\alpha_0 = g^s$.

Protocol

- 1. Party i shares the value of f(i) by means of a random polynomial $f^{(i)}(x)$ of order k'(k' < k) such that $f^{(i)}(0) = f(i)$. Then she broadcasts $\beta_j^{(i)} = g^{f_j^{(i)}}$ for $0 \le j \le k$, where $f_j^{(i)}$ is the j-th coefficient of the $f^{(i)}(x)$, as public checking parts for VSS. Note that $\beta_0^{(i)} = g^{f^{(i)}(0)} = g^{f(i)}$, which is publicly computable from the initial input of α_j for $0 \le j \le k$.
- 2. Party j receives the share $f^{(i)}(j)$ from i, and then announces i-correct or i-wrong according to whether the equation

$$g^{f^{(i)}(j)} \stackrel{?}{=} \prod_{m=0}^{k} \beta_m^{(i)}^{j^m} \pmod{p}$$

holds or not.

3. A set Λ of good parties of cardinality k+1 is defined as one in which the parties have more than k *i*-corrects. (If more than k+1 parties have more than k *i*-corrects, those with smaller id numbers are chosen.) Party j then computes $\sum_{i\in\Lambda} \lambda_i f^{(i)}(j)$, which results in the sharing of a polynomial

 $f'(x) = \sum_{i \in A} \lambda_i f^{(i)}(x)$ of order k'. Public values for VSS checking are also computed as follows:

$$\alpha'_j = g^{\sum_{i \in A} \lambda_i f_j^{(i)}} = \prod_{i \in A} (\beta_j^{(i)})^{\lambda_i}, \ for \ 0 \le j \le k'.$$

Lemma 3.2 The protocol 3.2 has the following properties:

(Robustness) The new shares computed at the end of the protocol correspond to the secret s. (That is, any subset of k'+1 of the new shares can reconstruct the secret s, and an adversary who can control up to k' parties cannot alter the result.)

(Privacy) Apart from the value g^s , an adversary who can control up to k' cannot learn anything about the secret.

(Freshness) The new share of each party is independent of its old share.

Sketch of Proof (Robustness) We can assume that there are k+1 > k' honest parties. Thus at least k+1 parties are announced as *i*-correct by k+1 parties. Thus the honest parties can compute the new share. By VSS, the party *i* can check that the share from party *j* lies on a polynomial which represents party *i*'s original share. It can also confirm that the new secret is the same as the original secret by comparing α_0 and α'_0 .

(Privacy) We can construct a simulator that, given the input of the corrupted parties, simulates the process of local secret sharing by party j. We can assume that party j is honest and that parties i for $1 \le i \le k'$ are corrupted. Using the k' shares $f^{(j)}(i)$ for $1 \le i \le k'$, together with the secret s, the simulator can set k'+1 equations and determine the polynomial of order k'+1. It then distributes the shares and VSS checking parts that have the same distribution as in the real protocol.

(Freshness) Party j's new share is computed as $\sum_{i \in \Lambda} \lambda_i f^{(i)}(j)$, where $f^{(i)}(j)$ is generated by party i by using random coefficients.

3.3 Robust Sequential Multiplication

The difficulty in applying Feldman's VSS to multiplication is that we cannot compute $g^{a \cdot b}$ from g^a and g^b . Thus, in some protocol [GRR98], the parties need to prove that the new published commitment $g^{a_i \cdot b_i}$ is really obtained from $(g^{a_i})^{b_i}$ (or $(g^{b_i})^{a_i}$), by using the zero-knowledge proof of equality of discrete-logs [Cha90], which proves that $DL_g(g^{b_i}) = DL_{g^{a_i}}(g^{a_i \cdot b_i})$ without requiring any information about b_i to be provided. However, the above ZKIP requires a verifier and interaction between the parties and a verifier, which reduces the advantages of non-interactive verification by Feldman's VSS.

Cerecedo et al. [CMI93] showed that VSS can be applied in a special case of multiplication where one of the multipliers is generated by the joint-shared-secret protocol. We will first review this protocol before proposing the general multiplication protocol:

Let $f(x)=a_0+\cdots+a_kx^k$ be a polynomial of order k used to share the value $a=a_0$. Party i holds the value f(i) mod q, and the values $\alpha_m=g^{a_m}$ mod p for $0\leq m\leq k$ are publicly known. She then generates $r^{(i)}(x)=r_0^{(i)}+\cdots+r_k^{(i)}x^k$ of order k and $z^{(i)}(x)=z_1^{(i)}x+\cdots+z_{2k}^{(i)}x^{2k}$ of order 2k for Joint-Random-VSS and Joint-Zero-VSS (see appendix A.4), respectively, which means that party j holds values $r^{(i)}(j), z^{(i)}(j)$ and the values $\beta_m=g^{r_m^{(i)}}, (0\leq m\leq k)$ and $\gamma_m=g^{z_m^{(i)}}, (0\leq m\leq 2k)$ are publicly known. Now let $y^{(i)}(x)=f(x)r^{(i)}(x)+z^{(i)}(x)$ be a polynomial of order 2k that is used to share the value $a\cdot r_0^{(i)}$.

The problem is how party i can validate the shares $y^{(i)}(j)$ that she distributes to party j. Let $y^{(i)}(x) = y_0^{(i)} + y_1^{(i)}x + \cdots + y_{2k}^{(i)}x^{2k}$; then public commitment values for VSS checking are:

$$\begin{split} & \delta_0^{(i)} = g^{y_0^{(i)}} = g^{a_0 \cdot r_0^{(i)}} = \alpha_0^{r_0^{(i)}} \pmod{p}, \\ & \delta_1^{(i)} = g^{y_1^{(i)}} = g^{a_0 \cdot r_1^{(i)} + a_1 \cdot r_0^{(i)} + z_1^{(i)}} = \alpha_0^{r_1^{(i)}} \alpha_1^{r_0^{(i)}} g^{z_1^{(i)}} \pmod{p}, \\ & \vdots \\ & \delta_{2k}^{(i)} = g^{y_{2k}^{(i)}} = g^{a_k \cdot r_k^{(i)} + z_2 k^{(i)}} = \alpha_k^{r_k^{(i)}} g^{z_{2k}^{(i)}} \pmod{p}, \end{split}$$

all of which can be locally computed by i and published. Thus party j can verify his share $y^{(i)}(j) = f(j)r^{(i)}(j) + z^{(i)}(j)$ by computing

$$g^{(f(j)r^{(i)}(j)+z^{(i)}(j))} \stackrel{?}{=} \prod_{m=0}^{2k} \delta_m^{(i)j^m} \pmod{p}.$$

All the verified parties' shares are then summed up to share the product $r \cdot a = \sum_{i \in \Lambda} r^{(i)} \cdot a$, where Λ denotes the set of qualified parties, and its public VSS commitment values are computed as follows:

$$\delta_j = g^{\sum_{i \in \Lambda} y_j^{(i)}} = \prod_{i \in \Lambda} \delta_j^{(i)}, \ for \ 0 \le j \le 2k.$$

Now, note that the result is a product of a and $r = \sum_{i \in \Lambda} r^{(i)}$, where $r^{(i)}$ is a random number generated by party i. But suppose that $r^{(i)}$ is the share of the polynomial h(x) of order 2k whose free coefficient is another secret s. Then s is recovered by Lagrange interpolation: $s = \sum_{i \in \Lambda} \lambda_i r^{(i)}$. This means that we can apply the above scheme to the multiplication of two jointly shared (not random) values. The protocol is defined as follows:

3.4 Protocol for Robust Sequential Multiplication

Input of Party i: f(i) of a polynomial f(x) of order k representing a secret a, and h(i) of a polynomial h(x) of order 2k representing a secret s.

Public Input: $\alpha_i = g^{f_i}$ for $0 \le i \le k$, and $\delta_i = g^{h_i}$ for $0 \le i \le 2k$, where f_i and h_i denote the *i*-th coefficients of f(x) and h(x), respectively.

Protocol

- 1. Using VSS, party i shares the value of h(i) by means of a random polynomial $h^{(i)}(x)$ of order k such that $h^{(i)}(0) = h(i)$. Note that $g^{h(i)}$ is already publicly computable from the public input δ .
- 2. Using VSS, party i shares 0 by means of a random polynomial $z^{(i)}(x)$ of order 2k such that $z^{(i)}(0) = 0$.
- 3. Party i broadcasts $\delta_j^{(i)} = g^{h_j^{(i)}}$ for $0 \le j \le 2k$, where $h_j^{(i)}$ denotes the j-th coefficient of a polynomial computed by $h^{(i)}(x) = f(x)h^{(i)}(x) + z^{(i)}(x)$.
- 4. Party j first checks the shares $h^{(i)}(j)$, $z^{(i)}(j)$ received from i, then checks the multiplication $f(j)h^{(i)}(j) + z^{(i)}(j)$ by confirming that $g^{(f(j)h^{(i)}(j)+z^{(i)}(j))} \stackrel{?}{=} \prod_{m=0}^{2k} \delta_m^{(i)^{j^m}} \pmod{p}$. If all the checks are validated, she announces i-correct; otherwise, she announces i-wrong.
- 5. Party i is included in a good set of parties Λ if more than 2k parties announced i-correct. Then, party j computes her share as $\sum_{i\in\Lambda}\lambda_i h^{(i)}(j)$, which results in sharing of a polynomial $h'(x)=\sum_{i\in\Lambda}\lambda_i h^{(i)}(x)$ of order 2k that carries the secret $h'(0)=a\cdot s$. Public values are also computed as follows:

$$\delta_j = g^{(\sum_{i \in \Lambda} \lambda_i h_j^{(i)})} = \prod_{i \in \Lambda} (\delta_j^{(i)})^{\lambda_i}, \text{ for } 0 \le j \le 2k.$$

Lemma 3.4 The above protocol is t-resilient for a malicious adversary when the total number of parties is $n \geq 3t + 1$. After the protocol has been carried out, each party's new share is independent of its old share.

Sketch of Proof (Robustness) We can assume that there are 2k+1 honest parties. Thus at least 2k+1 parties are announced to be *i*-correct by 2k+1 parties, and therefore the honest parties can compute the product.

(Privacy) We can construct a simulator that, given the input of the corrupted parties, simulates the process of local secret sharing. And the outputs have the same distribution as in the real protocol.

(Freshness) The proof is similar to that of Lemma 3.2.

4 $T \cup_{T_{PEG}}$ Based on PRSG and the ElGamal Encryption Scheme

Our goal is to implement function sharing of a pseudo-random sequence generator to obtain a periodical sequence of secrets, and to form a threshold cryptosystem using the secrets. We will use polynomial-based secret sharing over the ring Z_N , where N is the product of large primes. We assume that the number of parties n is smaller than any prime divisors of N, in order to apply the Lagrange interpolation formula. This assumption is not always satisfied in a linear congruence generator (LCG), and is thus an additional constraint.

4.1 $T \bigcup_{T_n} EG$ Protocol Based on LCG

Here we give an implementation of the periodical multi-secret threshold cryptosystem based on LCG-based key sequence generation (see appendix A.1) and the ElGamal encryption scheme.

It is well known that using an LCG as PRSG for the secret key is very dangerous, because it is possible to predict when some part of the sequence will become available. But in our application, the sequence is hidden in the exponent of the public key. Thus, because of the difficulty of calculating a discrete logarithm, the secret key is not predictable. Recently, the LCG used in the DSS has been attacked by solving the simultaneous modular linear equations obtained from two sets of DSS signatures [BGM97]. But such an attack cannot be used with our application.

Protocol for $T \bigcup_{T_v(LCG)} EG$ -KeyUpdate

Input to Party i $y^{[0]}(i)$ of polynomial $y^{[0]}(x)$ of order 2k representing a seed $s^{[0]}$, f(i) of polynomial f(x) of order k representing a, and h(i) of polynomial h(x) or order 2k representing b. A dealer is necessary to select an appropriate a, b, and seed $(s^{[0]})$, because the period of LCG depends on N, a, b, and $s^{[0]}$.

Public Input A composite number N and an element $g \in Z_p^*$ of order N, where all divisors of N are larger than the number of the parties (n). $\alpha = g^{f_i}$ for $0 \le i \le k$, where f_i denotes the i-th coefficient of f(x). $\beta_i = g^{h_i}$ and $\delta_i^{[0]} = g^{y_i^{[0]}}$ for $0 \le i \le 2k$, where h_i and $y_i^{[0]}$ denote the i-th coefficients of h(x) and $y^{[0]}(x)$, respectively.

Protocol

- 1. First, parties perform the robust sequential multiplication protocol (section 3.4) to multiply the polynomials $y^{[m]}(x)$ and f(x). Then they replace their share $y^{[m]}(i)$ by the result share of $s^{[m]} \cdot a$, which is represented by a polynomial of order 2k.
- 2. Then parties add h(i) to their intermediate results to obtain the shares of $a \cdot s^{[m]} + b$.
- 3. Finally, parties erase all the previous shares and intermediate results. The shares are carried by a polynomial $y^{[m+1]}(x)$ of order 2k. which represents a secret of $s^{[m+1]} \stackrel{def}{=} s^{[m]} \cdot a + b \pmod{N}$.

Protocol for *EG***-Encrypt** This protocol is common to all ElGamal-based encryption schemes: Since the public key $Y^{[m+1]} = g^{y_0^{[m+1]}} \mod p$ is publicly known, the user can encrypt his message without informing the key-generating parties. The message M is encrypted, by using random $K \in \mathbb{Z}_p$, as $(A, B) \stackrel{def}{=} (g^K, Y^{[m+1]}{}^K M) \pmod p$.

Protocol for $T \bigcup_{T_n(LCG)} EG$ -Decrypt

Input for all parties A ciphertext (A, B).

Protocol

- 1. Each party P_i sends to the receiver who is requesting decryption of the message the partial decryption $A_i = A^{y^{[m]}(i)} \mod p$ and proves to the receiver that $DLog_AA_i = DLog_gg^{y^{[m]}(i)}$ by using ZKIP [Cha90].
- 2. The receiver reconstructs the message by computing

$$M = \frac{B}{\prod_{i \in A} A_i^{\lambda_i}} \bmod p,$$

where Λ is a qualified set of parties of cardinality 2k+1.

Theorem 4.1 The above protocols are t-resilient for a mobile malicious adversary when the number of parties is $n \geq 3t + 1$.

Sketch of Proof We can assume that there are 2k + 1 honest parties.

(Robustness) The protocol for KeyUpdate consists of multiplication and addition of two shares. Lemma 3.4 is used for the VSS-supported multiplication, and the result is verifiably shared by 2k+1 honest parties. The robustness of the decryption protocol is obvious: unless a bad party passes the ZKIP with an incorrect partial decryption, the message is correctly decrypted.

(Privacy) Given the input of the corrupted parties as an adversary's view, we can construct a simulator that simulates the key update process and outputs an arbitrary Y as a public key with the same distribution as in the real protocol.

4.2 $T \bigcup_{T_n} EG$ Protocol Based on BBS-PRSG

Here we give another implementation of the periodical multi-secret threshold cryptosystem based on BBS-based key sequence generation (see appendix A.1) and the ElGamal encryption scheme.

The BBS is a secure bit generator when used as the PRSG, but here we will use the whole value, which like LCGs, is also predictable if some part of the sequence is available. But again, in our application, the values are hidden in the exponent of the public key. Now the question is, when an adversary sees the sequence: $g^y, g^{y^2}, \ldots, g^{y^{2^i}}$, can he predict the next sequence? The decisional Diffie-Helman (DDH) assumption says that, given $\langle P, Q, g, g^a, g^b \rangle$, there is no efficient algorithm that distinguishes g^c and $g^{a \cdot b}$. Thus, applying the DDH assumption where a = b(=y), the adversary cannot have any information about the next sequence.

Protocol for $T \bigcup_{T_n(BBS)} EG$ -KeyUpdate

Input to Party $i\ y^{[0]}(i)$ of polynomial $y^{[0]}(x)$ of order 2k representing a seed $s^{[0]}$. A dealer is necessary to select an appropriate seed $(s^{[0]})$, because the period of BBS depends on N and $s^{[0]}$.

Public Input A composite number N that is a product of two primes and an element $g \in Z_p^*$ of order N, where any divisors of N are larger than the number of the parties (n). $\delta_i^{[0]} = g^{y_i^{[0]}}$ for $0 \le i \le 2k$, where $y_i^{[0]}$ denotes the i-th coefficient of $y^{[0]}(x)$.

Protocol

- 1. The parties first perform the robust degree reduction protocol (section 3.2) to reduce the degree of $y^{[m]}(x)$ from 2k to k. The resulting polynomial is set to $y'^{[m]}(x)$.
- 2. The parties then perform the robust sequential multiplication protocol (section 3.4) for multiplying the polynomials $y'^{[m]}(x)$ and $y^{[m]}(x)$ in order to share $s^{[m]^2} \mod N$.
- 3. The parties finally erase all the previous shares and intermediate results. The shares are carried by a polynomial $y^{[m+1]}(x)$ of order 2k. which represents a secret of $s^{[m+1]} \stackrel{def}{=} s^{[m]^2} \pmod{N}$.

Protocol for $T \bigcup_{T_p(BBS)} EG$ -Decrypt The protocol is the same as $T \bigcup_{T_p(LCG)} EG$ -Decrypt (section 4.1).

Theorem 4.2 The above protocols are t-resilient for a mobile malicious adversary when the number of parties is $n \ge 3t + 1$.

The proof is similar to that for Theorem 4.1.

5 Conclusions

In this paper we have defined the periodical multi-secret threshold cryptosystem and given an implementation that consists of (1) a t-resilient key sequence generation and (2) a t-resilient encryption/decryption scheme. For (1), we designed a protocol for sharing a pseudo-random sequence generation function based on Feldman's VSS. We also gave a robust protocol for sequential multiplication. For (2), we used the ElGamal encryption scheme, which makes it easy to collectively generate a public key while the private key is implicitly maintained.

There is an opinion that from the viewpoint of trust relationships, once the user relies on distributed servers to enforce the timing of a decryption, timing can be also operationally maintained by the servers. But we argue that there is a big difference between the operationally maintained timing and the timing kept by the key refreshment process, because we can apply security theory (resiliency and

proactiveness) to the latter, while for the former the security is also maintained operationally.

By using different schemes for (1) and (2) and combining them, we can define various types of cryptographic system. For example, if we replace the encryption scheme by a signature scheme in (2), we can construct a time-restricted signature scheme, where the user has to visit more than k servers in a timeframe to get a signature. This might be used for contents metering.

We can also replace the PR-sequence generator with another sequence generator to control the period more easily. Actually, the BBS and linear congruential generators have a problem in that their period is too long, since it is dependent on the modulus, which should be large in order to maintain security. Thus we need to find an appropriate sequence generator whose period is relatively short and easy to control, yet whose sequence cannot be predicted by a malicious adversary.

Although we assume that the parameters and seed for the PRSG are fed by a dealer, we can eliminate the dealer by using Joint-Random-VSS to generate such values. But to realize this, we should consider the distributed checking mechanism for the period of the sequence.

References

- BGM97. Bellare, M., Goldwasser, S., and Micciancio, D., "Pseudo-Random Number Generation within Cryptographic Algorithms: the DSS Case," Advances in Cryptology - CRYTO'97, Lecture Notes in Computer Science 1294, Springer-Verlag, 1997. 371
- BBS86. Blum, L., Blum, M., and Shub, M., "A Simple Unpredictable Pseudo-random Number Generator," SIAM Journal on Computing, Vol. 15, No. 2, pp.364-383, 1986. 376
- BGW88. Ben-Or, M., Goldwasser, S., and Wigderson, A., "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation," Proceedings of the 20th ACM Symposium on Theory of Computing, pp.1-10, 1988. 366, 367, 377
- CCD88. Chaum, D., Crepeau, C. and Damgård, I., "Multiparty Unconditionally Secure Protocols," Proceedings of 20th ACM Symposium on Theory of Computing, pp.11-19, 1988. 366
- CH94. Canetti, R. and Herzberg, A., "Maintaining Security in the Presence of Transient Faults," Advances in Cryptology CRYPTO'94, Lecture Notes in Computer Science 839, Springer-Verlag, pp.425-438, 1994. 364
- Cha90. Chaum, D., "Zero-Knowledge Undeniable Signature," Advances in Cryptology EUROCRYPTO'90, Lecture Notes in Computer Science 473, Springer-Verlag, pp.458-464, 1991. 368, 372
- CMI93. Cerecedo, M., Matsumoto, T., and Imai, H., "Efficient and Secure Multiparty Generation of Digital Signatures Based on Discrete Logarithms," *IEICE Transaction on Fundamentals*, E76-A(4):522-533, 1993. 368
- Des88. Desmedt, Y., "Society and Group Oriented Cryptography: A New Concept," Advances in Cryptology - CRYPTO'87, Lecture Notes in Computer Science 293, Springer-Verlag, pp.120-127, 1988. 364

- DF90. Desmedt, Y. and Frankel, Y., "Threshold Cryptosystems," Advances in Cryptology CRYPTO'89, Lecture Notes in Computer Science 435, Springer-Verlag, pp.307-315, 1990. 364
- Fel87. Feldman, P, "A Practical Scheme for Non-interactive Verifiable Secret Sharing," Proceedings of the IEEE 28th Annual Symposium on Foundation of Computer Science, pp.427-437, 1987. 366, 376
- FGMY98. Frankel, Y., Gemmell, P., MacKenzie, P. D., and Yung, M., "Optimal-Resilience Proactive Public-Key Cryptosystems," *Proceedings of the IEEE 38th Annual Symposium on Foudation of Computer Sciences*, 1997. 364
- FY92. Franklin, M., and Yung, M., "Communication Complexity of Secure Computation," Proceedings of the 24th ACM Symposium on Theory of Computing, 1992. 364
- GRR98. Gennaro, R., O.Rabin, M., and Rabin, T., "Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography," Proceedings of the 17th ACM Symposium on Principles of Distributed Computing, 1998. 368
- HJKY95. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M., "Proactive Secret Sharing or: How to Copy With Perpetual Leakage," Advances in Cryptology CRYPTO'95, Lecture Notes in Computer Science 963, Springer-Verlag, pp.339-352, 1995. 364
- May
93. May T. C., "Timed-Release Crypto," Informal memo referred to by
 $[{\rm RSW96}].$ 363
- OY91. Ostrovsky, R. and Yung, M., "How to Withstand Mobile Virus Attacks," Proceedings of the 10th ACM Symposium on Principle of Distributed Computing, 1991. 364
- Sha79. Shamir, A., "How to Share A Secret," Communications of the ACM 22, 1979. 376
- Ped91a. Pedersen, T., "Distributed Provers with Applications to Undeniable Signatures," Advances in Cryptology EUROCRYPT'91, Lecture Notes in Computer Science 547, Springer-Verlag, 1991. 377
- Ped91b. Pedersen, T., "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," Advances in Cryptology - CRYPTO'91, Lecture Notes in Computer Science 576, Springer-Verlag, pp.129-140, 1991. 377
- Ped
91c. Pedersen, T., "A Threshold Cryptosystem without a Trusted Party," Advances in Cryptology EUROCRYPT'91, Lecture Notes in Computer Science 547, Springer-Verlag, pp.522-526, 1991.
- RSW96. Rivest, L.R., Shamir, A., and Wagner, D. A., "Time-Lock Puzzles and Time-Released Crypto," MIT Technical Paper, 1996. 363, 375

A Basic Tools

Here, we describe some of the existing tools that we use in our protocol.

A.1 Pseudo-random Sequence Generators (PRSG)

A random number sequence generator is defined by the form

$$X_n = f(X_{n-1}).$$

Here we will consider two well-known types of generators, linear congruential generators and BBS generators.

Linear Congruential Generators (LCG) A linear congruential generator is defined by the form

$$X_i = (aX_{i-1} + b) \bmod N,$$

where the parameters a, b, and N are constants. If they are properly chosen, the generator has an (N-1) period, which we call the maximal period generator.

Blum-Blum-Shub (BBS) Generators [BBS86] A BBS generator is defined by the form

$$X_i = X_{i-1}^2 \bmod N,$$

where modulus N is the product of two large prime numbers, p and q, that are congruent to $3 \mod 4$ (N is a Blum integer). If we specify the additional properties that $p_1 = (p-1)/2$, $p_2 = (p-3)/4$, $q_1 = (q-1)/2$, and $q_2 = (q-3)/4$ are all primes, then the period is a divisor of $2p_2q_2$.

A.2 Secret Sharing by Polynomial Interpolation [Sha79]

Shamir's secret sharing realized a (k,n)-threshold scheme according to which any k-1 parties have no information about the secret, while k can recover the secret. Suppose the secret to be shared is $s \in \mathbb{Z}_q$. The dealer generates a polynomial of order k-1 by randomly choosing its coefficients a_1, \ldots, a_n and sets the secret to its free coefficient $a_0 = s$:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_{k-1} x^{k-1}$$
.

The dealer then distributes the share $s_i = f(i) \mod q$ to the party P_i , $(1 \le i \le n)$ via a secure channel. By using Lagrange interpolation, the set of shareholders Λ of cardinality k determines

$$f(x) = \sum_{i \in \Lambda} \lambda_i f(i), \quad where \quad \lambda_i = \prod_{j \in \Lambda, j \neq i} (x - j)(i - j)^{-1}$$

thus reconstructing the secret $s = a_0 = f(0)$.

A.3 Verifiable Secret Sharing by Feldman [Fel87]

The verifiable secret sharing (VSS) scheme enables the receivers of shares to check whether the dealer has distributed the correct shares. Feldman proposed an efficient non-interactive VSS scheme that uses a homomorphic encryption function: Assume that a secret space is defined over a prime field Z_q . When a dealer generates a polynomial of order k-1, he broadcasts the values $\alpha_i = g^{a_i} \mod p$, for $0 \le i \le n$, where p is a prime such that q|p-1 and $g \in \mathbb{Z}_p^*$ is an element of order q. The receiver i can check that his/her share s_i is really the value of f(i) by calculating

$$g^{s_i} \stackrel{?}{=} \prod_{j=0}^{k-1} \alpha_j^{i^j} \bmod p$$

If the number of corrupt parties (t) is less than a half of the total (n), where $n \geq 2t + 1$, then the incorrect shares are recoverable in a (t + 1, n)-threshold VSS scheme.

A.4 Joint Random/Zero Secret Sharing [BGW88][Ped91a][Ped91b]

Sometimes it is convenient to generate and share a random value without a dealer. This can be done by means of the following protocol. A zero secret can also be shared in the same way. First, each party acts as a dealer of a random local secret or zero: the party i randomly generates a polynomial $f_i(x)$ of order k-1 whose free coefficient is set to a random value r_i or zero, and distributes the value $f_i(j)$ to party P_j . Then, each party sums up its shares in order to share a new function $g(x) = \sum_i f_i(x)$ whose free coefficient is $\sum_i r_i$ or zero. This protocol can be assisted by Feldman's VSS, which enables every party to verify that its share is correct. We will refer to these protocols as Joint-Random-VSS and Joint-Zero-VSS.

A Signature Scheme with Message Recovery as Secure as Discrete Logarithm

Masayuki Abe and Tatsuaki Okamoto

NTT Laboratories 1-1 Hikari-no-oka, Yokosuka-shi, 239-0847 Japan {abe,okamoto}@isl.ntt.co.jp

Abstract. This paper, for the first time, presents a provably secure signature scheme with message recovery based on the (elliptic-curve) discrete logarithm. The proposed scheme can be proven to be secure in the strongest sense (i.e., existentially unforgeable against adaptively chosen message attacks) in the random oracle model under the (elliptic-curve) discrete logarithm assumption. We give the concrete analysis of the security reduction. When practical hash functions are used in place of truly random functions, the proposed scheme is almost as efficient as the (elliptic-curve) Schnorr signature scheme and the existing schemes with message recovery such as (elliptic-curve) Nyberg-Rueppel and Miyaji schemes.

1 Introduction

1.1 Background: Digital Signature Schemes with Message Recovery

A digital signature scheme with message recovery is useful for many applications in which small messages (e.g., around 100 bits) should be signed. For example, small messages including time, date and identifiers are signed in certified email services and time stamping services. In addition, as shown in [13,14,15], the benefits of the message recovery are: direct use in other schemes such as identity-based public-key systems or key agreement protocols and natural combination with El-Gamal type encryption (which may produce the so-called sign-encryption).

The existing digital signature schemes with message recovery are classified into two types: RSA based schemes (RSA type) and discrete logarithm based schemes (DL type), where an elliptic curve based signature scheme is one of the DL type. PSS-R [2] and ISO/IEC 9796-1,9796-2 are signature schemes with message recovery in the RSA type, and the Nyberg-Rueppel [13,14,15], and Miyaji [11] schemes are in the DL type.

Recently the security flaws of heuristically designed schemes such as PKCS#1 (Ver.1) and the above-mentioned RSA-based signatures with message recovery, CoronISO/IEC 9796-1 and 9796-2, have been found [3,5]. Therefore, the *provable security* even in the random oracle model is most desirable to assure the security of a practical scheme.

Among the existing signature schemes with message recovery, only the PSS-R scheme [2] is provably secure (existentially unforgeable against adaptively chosen message attacks) under reasonable assumptions (the RSA assumption and random oracle model). In other words, there exists no provably secure signature schemes with message recovery in the DL type (i.e., no elliptic curve based signature scheme with message recovery) even in the random oracle model.

Since the overhead (size) of a digital signature based on the integer factoring should be much larger than such a small message (e.g., 1024 bit signature is much larger than 100 bit message), an elliptic curve based signature scheme is more appropriate for applications with small messages because of its small signature and key sizes.

That is, the most appropriate signature schemes with message recovery should be elliptic curve based schemes, while there exists no *provably secure* elliptic curve based signature scheme with message recovery (even in the random oracle model).

1.2 Our Result

This paper solves this problem. That is, we, for the first time, present a provably secure (existentially unforgeable against adaptively chosen message attacks) DL type (e.g., elliptic curve based) signature schemes with message recovery under reasonable assumptions, the (elliptic curve) discrete logarithm assumption and random oracle model. We also give the concrete analysis of the reduction to prove the security of the proposed signature scheme.

When practical hash functions are used in place of truly random functions, the proposed scheme is almost as efficient as the (elliptic-curve) Schnorr signature scheme and the existing schemes with message recovery such as (elliptic-curve) Nyberg-Rueppel and Miyaji schemes.

1.3 Related Works

Although Miyaji claimed that her scheme [11] is as secure as the elliptic curve DSA, the security level investigated in [11] is the weakest (i.e., universally unforgeable against passive attacks) and it is unlikely that her scheme is provably secure in a stronger security definition. Note that the security of signature schemes should be investigated based on the strongest security definition (i.e., existentially unforgeable against adaptively chosen message attacks) [9], in order to ensure the security against various possible attacks.

Remark: Recently Canetti et al. [4] have demonstrated that it is possible to devise cryptographic protocols which are provably secure in the random oracle model but for which no complexity assumption property instantiates the random oracle modeled hash function. However, the examples they used to make the random oracle model paradigm fail were very contrived, so the concerns induced by these examples do not appear to apply any of the concrete practical schemes that have been proven secure in the random oracle model.

2 Proposed Scheme

This section introduces our signature scheme with message recovery. Although we can construct our scheme based on any finite group, as a typical example, we will present a construction on the group over an elliptic curve because of its efficiency.

Key generation: Each signer S generates elliptic curve parameters, q for a finite field \mathbf{F}_q ; two elliptic curve coefficients a and b, elements of \mathbf{F}_q , that defines an elliptic curve E; a positive prime integer p dividing the number of points on E; and a curve point G of order p. Here |p| = k, and set (k_1, k_2) such that $|q| = k_1 + k_2$.

Signer S uniformly selects $x \in \mathbf{Z}/p\mathbf{Z}$, and calculates a point, Y, on E, where $Y = -x \cdot G$. The secret key of the signer is x, and its public-key is (\mathbf{F}_q, E, G, Y) .

The parameters, (\mathbf{F}_q, E, G) , of the elliptic curve domain can be fixed by the system and shared by many signers.

(In this paper, we follow the standard notations on the elliptic curve operation: the elliptic curve addition by +, and $G + \cdots + G$ (x times additions) by $x \cdot G$.)

Signature generation: S generates the signature, (r, z), of his message $m \in \{0, 1\}^{k_2}$ using public random oracle functions, $F_1 : \{0, 1\}^{k_2} \to \{0, 1\}^{k_1}$, $F_2 : \{0, 1\}^{k_1} \to \{0, 1\}^{k_2}$, $H : \{0, 1\}^{k_1 + k_2} \to \{0, 1\}^k$ as follows:

$$m' = F_1(m)||(F_2(F_1(m)) \oplus m),$$

$$r = (\omega \cdot G)_X \oplus m',$$

$$c = H(r),$$

$$z = \omega + cx \mod p,$$

where $\omega \in \mathbf{Z}/p\mathbf{Z}$ is uniformly selected. S sends (r,z) to verifier V. Here, P_X denotes the X-coordinate of point P on E, and \oplus denotes the bit-wise exclusive-or operation.

Verification: Verifier V recovers the message m from signature (r, z), and checks its validity as follows:

$$m' = r \oplus (z \cdot G + c \cdot Y)_X,$$

$$m = [m']_{k_2} \oplus F_2([m']^{k_1}),$$

and check whether $[m']^{k_1} = F_1(m)$ holds. Here, $[m']^{k_1}$ denotes the most significant k_1 bits of m', and $[m']_{k_2}$ denotes the least significant k_2 bits of m'.

Remark 1: If (r, z) is correctly generated, m should be recovered correctly and V accepts (r, z) as valid since

$$z \cdot G + c \cdot Y = \omega \cdot G + (cx) \cdot G - (cx) \cdot G = \omega \cdot G.$$

Remark 2: A typical security parameters for the signature scheme are: k = |p| = 160, |q| = 160, $k_1 = k_2 = 80$. Then, the message size, |m|, is 80 bits.

Remark 3: In order to sign a longer message (e.g., |m| > 80) with a fixed size of parameters of elliptic curve E (e.g., k = |p| = 160, |q| = 160, $|k_1| = |k_2| = 80$), m should be divided into two parts, m_1 and m_2 and $|m_1| = k_1$ (e.g., $|m_1| = 80$). Then, signer S generates (r, z) as follows:

$$m' = F_1(m_1)||(F_2(F_1(m_1)) \oplus m_1),$$

$$r = (\omega \cdot G)_X \oplus m',$$

$$c = H(r, m_2),$$

$$z = \omega + cx \mod p.$$

Signer S sends (r, z, m_2) to verifier V. V recovers m_1 from (r, z, m_2) as follows:

$$m' = r \oplus (z \cdot G + H(r, m_2) \cdot Y)_X,$$

$$m_1 = [m']_{k_2} \oplus F_2([m']^{k_1}),$$

and checks whether $[m']^{k_1} = F_1(m_1)$ holds.

Remark 4: The essential part in designing the signature scheme is how to construct the redundancy coding, m', of message m. Our coding is based on random functions (oracles), so that m' distributes uniformly over the randomness of the random functions, regardless of the distribution of m. The property based on random functions is used in the proofs of Lemmas 8 and 12. The property on the uniform distribution of m' is used in the proof of Lemma 12 (especially for Case 2).

3 Security

This section proves that the proposed signature scheme with message recovery is existentially unforgeable against adaptively chosen message attacks under the (elliptic-curve) discrete logarithm assumption and the random oracle model.

We will follow the proof methodology, the ID-reduction technique, introduced by [16] to analyze the reduction cost.

3.1 Security Definition of the Signature Scheme

We will quantify the security of a signature scheme: Here we assume that the attacker can dynamically ask the legitimate user S to sign any message, m, using him as a kind of oracle. This model covers the very general attack of the signature situations, adaptively chosen message attacks.

Definition 1. A probabilistic Turing machine (adversary) A breaks the proposed signature scheme with $(t, q_{sig}, q_{F_1}, q_{F_2}, q_H, \epsilon)$ if and only if A can forge a signature of a message with success probability greater than ϵ . We allow chosen-message attacks in which A can see up to q_{sig} legitimate chosen signatures participating in the signature generating procedure, and allow $q_{F_1}/q_{F_2}/q_H$ invocations of $F_1/F_2/H$, within processing time t. The probability is taken over the coin flips of A, F_1, F_2, H and signing oracle S.

Definition 2. The proposed signature scheme is $(t, q_{sig}, q_{F_1}, q_{F_2}, q_H, \epsilon)$ -secure if and only if there is no adversary that can break it with $(t, q_{sig}, q_{F_1}, q_{F_2}, q_H, \epsilon)$.

3.2 Intractability Definition of the Elliptic Curve Discrete Logarithm Problem

Definition 3. A probabilistic Turing machine (adversary) A breaks the elliptic curve discrete logarithm problem, (\mathbf{F}_q, E, G, Y) , with (t, ϵ) if and only if A can find x from (\mathbf{F}_q, E, G, Y) with success probability greater than ϵ within processing time t, where $Y = x \cdot G$. The probability is taken over the coin flips of A. Here, \mathbf{F}_q denotes a finite field with q elements, E denotes an elliptic curve over \mathbf{F}_q , and G is a point of E with prime order p.

Definition 4. The elliptic curve discrete logarithm problem, (\mathbf{F}_q, E, G, Y) , is (t, ϵ) -secure if and only if there is no adversary that can break it with (t, ϵ) .

3.3 Identification Scheme Induced from our Signature Scheme

Here we introduce the identification scheme that is induced from the abovementioned signature scheme. This identification scheme is useful to analyze the concrete security of our signature scheme, since the *ID Reduction Technique* in [16] with using this induced identification scheme is very effective for the security analysis.

In the identification scheme, prover P publishes a public key while keeping the corresponding secret key, and proves his identity to verifier V. Here, functions F_1 and F_2 are shared by P and V.

(Identification Scheme)

Key generation: Prover P generates a pair of a secret key, x, and a public key, (\mathbf{F}_q, E, G, Y) , using a key generation algorithm \mathcal{G} , where $Y = -x \cdot G$.

Identification Protocol: P proves his identity, and verifier V checks the validity of P's proof as follows:

-P selects m and generates r as follows:

$$m' = F_1(m)||(F_2(F_1(m)) \oplus m),$$

$$r = (\omega \cdot G)_X \oplus m',$$

where $\omega \in \mathbb{Z}/p\mathbb{Z}$ is uniformly selected. S sends r to verifier V.

- V generates random challenge $c \in \{0,1\}^k$ and sends it to P.
- -P generates an answer z as follows:

$$z = \omega + cx \mod p$$
.

P sends z to V

- V checks the validity of (r, z) through whether $[m']^{k_1} = F_1(m)$ holds or not, where

$$m' = r \oplus (z \cdot G + c \cdot Y)_X,$$

$$m = [m']_{k_2} \oplus F_2([m']^{k_1}).$$

3.4 Security Definition of the Identification Scheme

Definition 5. A probabilistic Turing machine (adversary) A breaks an identification scheme with $(t, q_{F_1}, q_{F_2}, \epsilon)$ if and only if A as a prover can cheat honest verifier V with a success probability greater than ϵ within processing time t. A is allowed to make q_{F_1} (and q_{F_2}) invocations of F_1 (and F_2). Here, the probability is taken over the coin flips of A, F_1 , F_2 and V.

Definition 6. An identification scheme is $(t, q_{F_1}, q_{F_2}, \epsilon)$ -secure if and only if there is no adversary that can break it with $(t, q_{F_1}, q_{F_2}, \epsilon)$.

3.5 ID Reduction Lemmas of the Proposed Signature Scheme

ID Reduction Technique introduced by [16] is effective to analyze the security of a certain class of signature schemes.

We can straightforwardly obtain the following lemma from the corresponding lemma in [16].

Lemma 7. (ID Reduction Lemma)

- 1) If A_1 breaks the proposed signature scheme with $(t, q_{sig}, q_{F_1}, q_{F_2}, q_H, \epsilon)$, there exists A_2 which breaks the signature scheme with $(t, q_{sig}, q_{F_1}, q_{F_2}, 1, \epsilon')$, where $\epsilon' = \frac{\epsilon \frac{1}{2k}}{q_H}$.
- 2) If \hat{A}_2 breaks the proposed signature scheme with $(t, q_{sig}, q_{F_1}, q_{F_2}, 1, \epsilon')$, there exists A_3 which breaks the signature scheme with $(t', 0, q_{F_1}, q_{F_2}, 1, \epsilon'')$, where $\epsilon'' = \epsilon' \frac{q_{sig}}{2^k}$ and $t' = t + (the simulation time of <math>q_{sig}$ signatures).
- 3) If A_3 breaks the proposed signature scheme with $(t', 0, q_{F_1}, q_{F_2}, 1, \epsilon'')$, there exists A_4 which breaks the induced identification scheme with $(t', q_{F_1}, q_{F_2}, \epsilon'')$

We neglect the time of reading/writing data on (random, communication, etc.) tapes, simple counting, and if-then-else controls. (Hereafter in this paper, we assume them.)

To analyze our scheme, the following lemma is additionally required, since random oracles F_1 and F_2 are used and shared by P and V.

Lemma 8. (Additional Reduction Lemma)

If A_4 breaks the identification scheme with $(t', q_{F_1}, q_{F_2}, \epsilon'')$, there exists A_5 which breaks the identification scheme with $(t', 1, 1, \epsilon''')$, where $\epsilon''' = \frac{1}{q_{F_1}} (\epsilon'' - \max\{\frac{1}{2^{k_1}}, \frac{1}{2^{k_2}}\})$.

Proof. Let Q_i be the *i*-th query from A_4 to random oracle F_1 and ρ_i be the *i*-th answer from F_1 to A_4 . Let $R_j = \rho_i$ be the query from A_4 to random oracle F_2 , which is consistent with Q_i .

Construct A_5 using A_4 as follows:

- 1. Select integer i with $1 \le i \le q_{F_1}$ randomly.
- 2. Run A_4 with random oracles, F_1 and F_2 , and a random working tape, Θ , where only the *i*-th query, Q_i , to F_1 and the related consistent query, R_j , to F_2 are asked to F_1 and F_2 , and the remaining $(q_{F_1} 1)$ queries to F_1 and $(q_{F_2} 1)$ queries to F_2 are asked to Θ . Here Θ contains $(q_{F_1} 1)$ k_2 -bitrandom-strings and $(q_{F_2} 1)$ k_1 -bit-random-strings used for answers from Θ .
- 3. Output the same as that of A_4 (i.e., A_5 succeeds if A_4 succeeds) if (r, z) output by A_4 satisfies the following:

$$m = Q_i, \quad [m']^{k_1} = R_j,$$

where

$$m' = r \oplus (z \cdot G + c \cdot Y)_X,$$

$$m = [m']_{k_2} \oplus F_2([m']^{k_1}).$$

Otherwise A_5 fails and halts.

If A_4 succeeds in making V accept (r, z) there are two cases: 1) m was asked to random oracle F_1 and $[m']^{k_1}$ was asked to random oracle F_2 , 2) otherwise.

In the latter case, the success probability of A_4 is at most $\max\{1/2^{k_1}, 1/2^{k_2}\}$, because of the randomness of the random oracle. Thus

 $Pr[A_5 \text{ succeeds}]$

$$\geq \sum_{i=1}^{q_{F_1}} \Pr[A_5 \text{ selects } i] \Pr[A_4 \text{ succeeds } \land (m = Q_i, [m']^{k_1} = R_j)]$$

$$= \sum_{i=1}^{q_{F_1}} \frac{1}{q_{F_1}} \Pr[A_4 \text{ succeeds } \land (m = Q_i, [m']^{k_1} = R_j)]$$

$$= \frac{1}{q_{F_1}} \sum_{i=1}^{q_{F_1}} \Pr[A_4 \text{ succeeds } \land (m = Q_i, [m']^{k_1} = R_j)]$$

$$= \frac{1}{q_{F_1}} (\Pr[A_4 \text{ succeeds}] - \Pr[A_4 \text{ succeeds } \land A_4 \text{ makes no query to } F_1 \text{ or } F_2])$$

$$\geq \frac{1}{q_{F_1}} (\epsilon'' - \max\{\frac{1}{2^{k_1}}, \frac{1}{2^{k_2}}\}),$$

because $\Pr[A_4 \text{ succeeds}] \ge \epsilon''$.

3.6 Security of the Induced Identification Scheme

A Boolean matrix and heavy row will be introduced in order to analyze the security of the above-mentioned identification scheme induced from the proposed signature scheme. Assume that there is a cheater A who can break a one-round identification scheme with $(t, 1, 1, \epsilon)$.

Here there are two cases: (Case 1) A's query to F_1 is made before sending r, and (Case 2) A's query to F_1 is made after sending r.

Let $\epsilon_1 + \epsilon_2 = \epsilon$, and A's success probability with Case 1 is at least ϵ_1 and A's success probability with Case 2 is at least ϵ_2 .

Definition 9. (Boolean Matrix of (A, V))

Let's consider the possible outcomes of the execution of (A, V) as a Boolean matrix $\mathcal{H}((RA, F_1, F_2), c)$ whose rows correspond to all possible choices of (RA, F_1, F_2) , where RA is a private random tape of A; its columns correspond to all possible choices of c, which means $c \in RV$, where RV is a random tape of V. Its entries are 0 if V rejects A's proof or V accepts A's proof with Case 2, and 1 if V accepts A's proof with Case 1.

Definition 10. (Heavy Row)

A row of matrix of \mathcal{H} is heavy if the fraction of 1's along the row is at least $\epsilon_1/2$, where the success probability of A with Case 1 is at least ϵ_1 .

Lemma 11. (Heavy Row Lemma)

The 1's in \mathcal{H} are located in heavy rows of \mathcal{H} with a probability of at least $\frac{1}{2}$.

Lemma 12. (Security of the identification scheme induced from the signature scheme)

Let $\epsilon \geq \frac{10}{2^k}$. Suppose that the elliptic curve discrete logarithm problem, (\mathbf{F}_q, E, G, Y) , is (t^*, ϵ^*) -secure. Then the identification scheme induced from the signature scheme is $(t, 1, 1, \epsilon)$ -secure, where

$$t^* = \frac{6(t + \Phi_1)}{\epsilon - 2/p} + \Phi_3$$
 and $\epsilon^* = \frac{1}{2} \left(1 - \frac{1}{e} \right)^2 > \frac{9}{50}$.

Here Φ_1 is the verification time of the identification protocol, Φ_3 is the calculation time of x in the final stage of the reduction, and e is the base of the natural logarithm.

Proof. Assume that there is a cheater A who can break an identification with $(t, 1, 1, \epsilon)$. We will construct a machine A^* which breaks the elliptic curve discrete logarithm problem, (\mathbf{F}_q, E, G, Y) , with (t^*, ϵ^*) using A.

First, we assume that $\epsilon_1 \geq \epsilon/2$, where either case occurs, $\epsilon_1 \geq \epsilon/2$ or $\epsilon_2 > \epsilon/2$, since $\epsilon_1 + \epsilon_2 = \epsilon$. (Later we consider the case when $\epsilon_2 > \epsilon/2$.)

We will discuss the following probing strategy of \mathcal{H} to find two 1's along the same row in \mathcal{H} [8]:

- 1. Probe random entries in \mathcal{H} to find an entry $a^{(0)}$ with 1. Let $c^{(0)}$ be V's challenge message corresponding to $a^{(0)}$. We denote the row where $a^{(0)}$ is located in \mathcal{H} by $\mathcal{H}^{(0)}$.
- 2. After $a^{(0)}$ is found, probe random entries along $\mathcal{H}^{(0)}$ to find another entry with 1. We denote it by $a^{(1)}$ and $c^{(1)}$ is V's challenge message corresponding to $a^{(1)}$. If $c^{(1)} \equiv -c^{(0)} \pmod{p}$, then discard it and find another entry with 1.

 $a^{(i)}$ represents $(r^{(i)}, z^{(i)})$. Here, $[m^{(i)'}]^{k_1} = F_1(m^{(i)})$ holds, since $a^{(i)}$ is an entry with 1, where

$$m^{(i)\prime} = r^{(i)} \oplus (z^{(i)} \cdot G + c^{(i)} \cdot Y)_X,$$

$$m^{(i)} = [m^{(i)\prime}]_{k_2} \oplus F_2([m^{(i)\prime}]^{k_1}).$$

Two 1's, $a^{(0)}$ and $a^{(1)}$, in the same row $\mathcal{H}^{(0)}$ means $r^{(1)} = r^{(0)}$, $m^{(1)'} = m^{(0)'}$, and $m^{(1)} = m^{(0)}$. Therefore,

$$(z^{(1)} \cdot G + c^{(1)} \cdot Y)_X = (z^{(0)} \cdot G + c^{(0)} \cdot Y)_X,$$

where $c^{(0)} \neq c^{(1)}$. That is,

$$z^{(1)} \cdot G + c^{(1)} \cdot Y = \pm (z^{(0)} \cdot G + c^{(0)} \cdot Y).$$

Hence if this strategy succeeds, x with $Y = x \cdot G$ can be computed by

$$x = -\frac{z^{(1)} \mp z^{(0)}}{c^{(1)} \mp c^{(0)}} \bmod p,$$

since p is prime and $c^{(1)} / \equiv e^{(0)} \pmod{p}$.

Then we will show that this strategy succeeds with constant probability in just $O(1/\epsilon_1)$ probes, using Lemma 11 concerning a useful concept, heavy row, defined in Definition 10.

Let p_1 be the success probability of step 1 with $\frac{1}{\epsilon_1}$ repetition. $p_1 \geq 1 - (1 - \epsilon_1)^{1/\epsilon_1} = p_1' > 1 - \frac{1}{e} > \frac{3}{5}$, because the fraction of 1's in \mathcal{H} is at least ϵ_1 . Let p_2 be the success probability of step 2 with $\frac{2}{\epsilon_1 - 1/p}$ repetition. $p_2 \geq \frac{1}{2} \times \left(1 - (1 - \frac{\epsilon_1 - 1/p}{2})^{2/(\epsilon_1 - 1/p)}\right) = p_2' > \frac{1}{2}(1 - \frac{1}{e}) > \frac{3}{10}$, because the probability that $\mathcal{H}^{(0)}$ is heavy is at least $\frac{1}{2}$ by Lemma 11 and the fraction of 1's (with $c^{(1)} / \equiv e^{(0)} \pmod{p}$) along a heavy row is at least $\frac{\epsilon_1 - 1/p}{2}$.

Let ϵ_1^* be the success probability of the above-mentioned procedure and t_1^* be the running time for procedure. Then

$$\begin{split} \epsilon_1^* &= p_1 \times p_2 \geq p_1' \times p_2' > \frac{1}{2} (1 - \frac{1}{e})^2 > \frac{9}{50}, \\ t_1^* &\leq (t + \varPhi_1) \times (\frac{1}{\epsilon_1} + \frac{2}{\epsilon_1 - 1/p}) + \varPhi_3 \\ &< \frac{3(t + \varPhi_1)}{\epsilon_1 - 1/p} + \varPhi_3 \end{split}$$

$$\leq \frac{6(t+\varPhi_1)}{\epsilon-2/p} + \varPhi_3.$$

Next, we consider the case when $\epsilon_2 > \epsilon/2$. Then A's success probability with Case 2 (A's query to F_1 is made after sending r) is greater than $\epsilon/2$.

Execute random trials of $((RA, F_1, F_2), c)$ to find a value of $((RA, F_1, F_2), c)$ in which A succeeds with Case 2. Here in each trial, the replies of F_1 and F_2 are set as follows: F_1 's reply: $[m']^{k_1}$, and F_2 's reply: $[m']_{k_2}$, where $m' = (\delta \cdot G)_X \oplus r$ and δ is uniformly selected from $\mathbb{Z}/p\mathbb{Z}$. Here note that although the distribution of m' is not guaranteed to be uniform (since $(\delta \cdot G)_X$ is not uniform), A succeeds with Case 2 only when the values of m' is in the distribution of $(\delta \cdot G)_X \oplus r$. Therefore, the success probability of A with Case 2 under the above-mentioned strategy of F_1 and F_2 is at least that under the uniform distribution of F_1 and F_2 (i.e., greater than $\epsilon/2$).

If a value of $((RA, F_1, F_2), c)$ in which A succeeds with Case 2 is found, x with $Y = x \cdot G$ can be computed by $x = (-z \pm \delta)/c \mod p$, since $(z \cdot G + c \cdot Y)_X = m' \oplus r = (\delta \cdot G)_X$.

Let ϵ_2^* be the success probability of the above-mentioned procedure and t_2^* be the running time for procedure.

$$\epsilon_2^* \ge 1 - (1 - \epsilon/2)^{2/\epsilon} = p_1' > 1 - \frac{1}{e} > \frac{3}{5},$$

$$t_2^* \le (t + \Phi_1) \times \frac{2}{\epsilon} + \Phi_3.$$

Since the first step in the probing stage with Case 1 and the probing stage with Case 2 can be merged as the unified probing stage, we can obtain the total success probability and running time as follows:

$$t^* = \frac{6(t + \Phi_1)}{\epsilon - 2/p} + \Phi_3$$
 and $\epsilon^* = \frac{1}{2} \left(1 - \frac{1}{e} \right)^2 > \frac{9}{50}$,

because $t_1^* > t_2^*$ and $\epsilon_1^* < \epsilon_2^*$.

3.7 Security of the Proposed Signature Scheme

The following theorem is proven by combining Lemmas 7, 8 and 12.

Theorem 13. (Security of the proposed signature scheme)

Let $\epsilon \geq q_H(\frac{10q_{F_1}+q_{sig}}{2^k}+\max\{\frac{1}{2^{k_1}},\frac{1}{2^{k_2}}\})+\frac{1}{2^k}$. Suppose that the elliptic curve discrete logarithm problem, (\mathbf{F}_q,E,G,Y) , is (t^*,ϵ^*) -secure. Then the proposed signature scheme with message recovery is $(t,q_{sig},q_{F_1},q_{F_2},q_H,\epsilon)$ -secure, where

$$t^* = \frac{6t'}{\epsilon''' - 2/p} + \Phi_3 \quad and \quad \epsilon^* = \frac{1}{2} \left(1 - \frac{1}{e} \right)^2 > \frac{9}{50}.$$

Here

$$t' = t + \Phi_1 + \Phi_2$$
 and $\epsilon''' = \frac{1}{q_{F_1}} \left(\frac{\epsilon - \frac{1}{2^k}}{q_H} - \frac{q_{sig}}{2^k} - \max\{\frac{1}{2^{k_1}}, \frac{1}{2^{k_2}}\} \right)$

where Φ_1 is the verification time of the identification protocol, Φ_2 is the simulation time of q_{sig} signatures, Φ_3 is the calculation time of x in the final stage of the reduction.

Remark: This theorem implies in an asymptotic sense that the proposed signature scheme with message recovery is existentially unforgeable against adaptively chosen massage attacks in the random oracle model, if the elliptic curve discrete logarithm (ECDL) problem, (\mathbf{F}_q, E, G, Y), is intractable. This is because: if ECDL is intractable (i.e., t^* is not within k^{c_1} for constant c_1 with constant ϵ^*), it is not true that t is within k^{c_2} for constant c_2 and ϵ is at least $\frac{1}{k^{c_3}}$ for constant c_3 , where $q_{sig}, q_{F_1}, q_{F_2}, q_H$ are at most polynomials in k, and $k_1 = c_4k$ and $k_2 = c_5k$ for constants c_4 and c_5 .

4 Conclusion

This paper presented a provably secure signature scheme with message recovery based on the (elliptic-curve) discrete logarithm. The proposed scheme is proven to be secure in the strongest sense (i.e., existentially unforgeable against adaptively chosen message attacks) in the random oracle model under the (elliptic-curve) discrete logarithm assumption. We provided the concrete analysis of the security reduction. When practical hash functions are used in place of truly random functions, the proposed scheme is almost as efficient as the (elliptic-curve) Schnorr signature scheme and the existing schemes with message recovery such as (elliptic-curve) Nyberg-Rueppel and Miyaji schemes (because the additional computation of our scheme compared with the Schnorr signature scheme is just the function evaluation of F_1 and F_2 , and data comparison).

References

- M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," Proc. of the First ACM Conference on Computer and Communications Security, pp.62–73, 1993.
- M. Bellare and P. Rogaway, "The Exact Security of Digital Signatures -How to Sign with RSA and Rabin," Proc. of Eurocrypt'96, Springer-Verlag, LNCS, pp.399-416, 1996. 378, 379
- D. Bleichenbacher, "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1," Proc. of Crypto'98, LNCS 1462, Springer-Verlag, pp. 1–12, 1998. 378
- R. Canetti, O. Goldreich and S. Halevi, "The Random Oracle Methodology, Revisited," Proc. of STOC, ACM Press, pp.209–218, 1998.
- J.S., D. Naccache and J.P. Stern, "On the Security of RSA Padding," Proc. of Crypto'99, Springer-Verlag, LNCS, 1999. 378
- T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions on Information Theory, IT-31, 4, pp.469

 472, 1985.
- A. Fiat and A. Shamir, "How to Prove Yourself," Proc. of Crypto'86, Springer-Verlag, LNCS, pp.186–194.

- 8. U. Feige, A. Fiat and A. Shamir, "Zero-Knowledge Proofs of Identity," J. of Cryptology, 1, p.77–94, 1988. 385
- S. Goldwasser, S. Micali and R. Rivest, "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," SIAM J. on Computing, 17, pp.281– 308, 1988.
- N. Koblitz, "Elliptic Curve Cryptosystems," Mathematics of Computation, 48, pp.203–209, 1987.
- A. Miyaji, "A Message Recovery Signature Scheme Equivalent to DSA over Elliptic Curves," Proc. of Asiacrypt'96, Springer-Verlag, LNCS, pp. 1–14, 1996. 378, 379
- 12. M. Naor and M. Yung, "Universal One-Way Hash Functions and Their Cryptographic Applications," Proc. of STOC, pp.33–43, 1989.
- K. Nyberg and R.A. Rueppel, "A New Signature Scheme Based on the DSA Giving Message Recovery," Proc. of the First ACM Conference on Computer and Communications Security, 1993. 378
- K. Nyberg and R.A. Rueppel, "Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem," Proc. of Eurocrypt'94, Springer-Verlag, LNCS, pp.182–193, 1995. 378
- K. Nyberg and R.A. Rueppel, "Message Recovery for Signature Schemes Based on the Discrete Logarithm Problem," Designs, Codes and Cryptography, 7, pp.61–81, 1996. 378
- K. Ohta and T. Okamoto, "On the Concrete Security Treatment of Signatures Derived from Identification," Proc. of Crypto'98, Springer-Verlag, LNCS, 1998. 381, 382, 383
- D. Pointcheval and J. Stern, "Security Proofs for Signature Schemes," Proc. of Eurocrypt'96, Springer-Verlag, LNCS, pp.387–398, 1996.
- 18. J. Rompel, "One-Way Functions are Necessary and Sufficient for Secure Signature," Proc. of STOC, pp.387–394, 1990.
- R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," Communications of ACM, 21, 2, pp.120-126, 1978.
- C.P. Schnorr, "Efficient Identification and Signatures for Smart Card," Proc. of Eurocrypt'89, Springer-Verlag, LNCS, pp.235–251, 1990.

A^3 -codes under Collusion Attacks

Yejing Wang and Rei Safavi-Naini

School of IT and CS, University of Wollongong, Northfields Ave., Wollongong 2522, Australia {yw17, rei}@uow.edu.au

Abstract. An A^3 -code is an extension of A-code in which none of the three participants, transmitter, receiver and arbiter, is assumed trusted. In this paper we extend the previous model of A^3 -codes by allowing transmitter and receiver not only to individually attack the system but also collude with the arbiter against the other. We derive information-theoretic lower bounds on success probability of various attacks, and combinatorial lower bounds on the size of key spaces. We also study combinatorial structure of optimal A^3 -code against collusion attacks and give a construction of an optimal code.

1 Introduction

Authentication codes (A-codes) [7] provide protection for two trustworthy participants against an active spoofer tampering with the messages sent by a transmitter to a receiver over a public channel. In this model transmitter and receiver are assumed trusted. An extension of this model is an authentication codes with arbitration [8], or A^2 -codes for short, in which transmitter and receiver are not trusted: transmitter may deny a message that he/she has sent, and receiver may attribute a fraudulent message to the transmitter. In an A^2 -code a trusted third party, called arbiter, resolves the dispute between transmitter and receiver. A^2 -codes have been studied by various authors [3], [4] and [6].

Brickell and Stinson ([1]) introduced authentication code with dishonest arbiter(s), or A^3 -code, in which the arbiter may tamper with the communication but it will remain trusted in her arbitration. In an A^3 -code each participant in the system has some secret key information which is used to protect him/her against attacks in the system. These codes have been also studied in [2], [3] and [9], where some constructions were given. However none of these constructions protect against collusion attacks.

Collusion attacks in A-codes are studied in various extensions of A-codes, such as multisender schemes [10] where unauthorised groups of senders can collude to construct a fraudulent message that is attributed to an authorised group, and multreceiver schemes where unauthorised groups of receivers collude to construct a fraudulent message that is attributed to the transmitter. The model studied in [3] is an extension of multreceiver schemes where transmitter can collude with unauthorised groups of receivers. In the former two cases no arbitration is required as at least one side in the communication is trusted, that is

receiver in a multisender scheme and transmitter in a multireceiver scheme are assumed trusted. However in the last case none of the sides is trusted and there can be a dispute between a receiver and a colluding group of the sender and receivers. The suggestion for resolving the dispute is to either include a trusted arbiter or, take the majority vote of the receivers.

In this paper we extend the attack model of A^3 -codes to include collusion between arbiter and transmitter or receiver, against the other participants. For example, the arbiter may collude with the transmitter to construct a message that the transmitter can later deny sending it, or she may collude with the receiver to impersonate the sender or substitute a message that he has sent. We assume that the arbiter always honestly follows the arbitration rules. These rules are public and collusion with a participant effectively means that the arbiter will make her key information available to that participant.

The paper is organised as follows. In Section 2 we introduce the model and derive information theoretic bounds on success probabilities of various attacks, and combinatorial bounds on the size of key spaces for each participant. Section 3 gives combinatorial structure of Cartesian optimal A^3 -codes. Section 4 gives a construction for such codes.

2 Model and Bounds

There are three participants: a transmitter, T, a receiver, R and an arbiter, A, none of them is assumed trusted. T wants to send a source state s, $s \in S$, to R over a public channel. Each participant has some secret key. T uses his key information to construct a message $m \in M$ for a source state s to be sent over the channel. R uses her key information to verify authenticity of a received message and finally A who does not know the key information of T and R will use her key information to resolve a dispute between the two. Transmitter's key, e_t , determines the encoding function $e_t: S \to M$ used by the transmitter, and receiver's key, e_r , determines the verification function $e_r: M \to S \cup \{F\}$ and so the subset of M that R will accept as valid message. Arbiter's key, e_a , determines a subset $M(e_a) \subset M$, which the arbiter accepts as valid.

There is also an *outsider*, O, who has no key information. A colluding group of attackers in general use their knowledge of the system, their key information and all the previous communicated messages to construct fraudulent messages.

The system has the following stages.

Key Distribution: during which a triple (e_t, e_r, e_a) of keys for the three participants T, R, and A is generated and each participant's key is securely delivered to him/her. This stage can be either performed by a trusted party, or by a collaboration of the three principals. A valid triple has the property that if $e_t(s) = m$ then $e_r(m) = s$ and $e_a(m) = s$.

Authentication: T uses e_t to generate an authentic message.

Verification: R uses e_r to verify authenticity of a received message m. She will also always ask for the verdict of A on the message. R will only accept a message if both R and A accept the message as authentic. We note that in [3] a

message is acceptable by the receiver if $e_r(m) \in S$. This means that $e_r(m) \in S$ implies $e_a(m) \in S$. We assume $M(e_r) \cap M(e_a) \neq \emptyset$ and so for every message both conditions, $e_r(m) \in S$ and $e_a(m) \in S$, must be checked.

Arbitration: A dispute occurs in a number of situations. In the following we list possible disputes and the rules for resolving the disputes. We note that the rules are public and A's arbitration can be later verified by everyone.

- 1. Arbitration Rule I (AR I): T denies sending a message m.
 - T wins if m is acceptable by e_r and e_a .
 - T looses otherwise.
- 2. Arbitration Rule II (AR II): R attributes a message m to T but T denies it.
 - R wins if m is valid under e_a .
 - -R looses otherwise.

The system is subject to the following attacks.

- **1. Attack** O_i : Observing a sequence of i legitimate messages m_1, m_2, \dots, m_i , the opponent places another message m into the channel. He is successful if both the receiver and the arbiter accept m as an authentic message.
- **2. Attack** R_i : Receiving a sequence of i legitimate messages, m_1, m_2, \dots, m_i , and using her key, e_r , R claims that she has received a message $m \neq m_1, m_2, \dots, m_i$. She is successful if A accepts m under the arbitration rule II.
- **3. Attack** A_i : Knowing a sequence of i legitimate messages m_1, m_2, \dots, m_i , and using her key e_a , the arbiter puts another message m into the channel. She is successful if the message is valid for R.
- **4. Attack** T_0 : Using his key (an encoding rule) e_t , transmitter sends a fraudulent message m which is not valid under e_t . He succeeds if both the receiver and the arbiter accept the message.
- **5.** Collusion Attack \overline{RA}_i : Having received a sequence of i legitimate messages m_1, m_2, \dots, m_i, R and A collude to construct a message which R claims it is sent by the transmitter. They succeed if m can be generated by the transmitter under e_t .
- **6. Collusion Attack** TA: A and T, using their keys e_t and e_a , collude to construct a message m which is not valid under e_t but using AR I makes T a winner.

Let E_T, E_R and E_A be the set of transmitter's, receiver's and arbiter's keys, respectively, and assume p(x, y, z) is the joint probability distribution on $E_T \times E_R \times E_A$. Also assume there is a probability distribution on the set of source states S. Denote the *support* of the joint probability distribution by

$$E_T \circ E_R \circ E_A = \{(e_t, e_r, e_a) : p(e_t, e_r, e_a) > 0\}.$$

The joint probability distribution determines the marginal distributions: $p(e_r, e_a)$, $p(e_t)$, $p(e_t)$, and $p(e_a)$. Similarly denote

$$E_R \circ E_A = \{(e_r, e_a) : p(e_r, e_a) > 0\}.$$

We will also use the following notations.

$$E_{\overline{RA}}(m^i) = \{(e_r, e_a) \in E_R \circ E_A : \text{ both } e_r, e_a \text{ accept } m^i\},\ E_{\overline{RA}}(e_t) = \{(e_r, e_a) \in E_R \circ E_A : p(e_t, e_r, e_a) > 0\}.$$

Let M be the set of all possible messages, and M^i denote the set of sequences of i distinct messages.

Using these notations, success probabilities in various attacks are given as follows.

$$P_{O_i} = \max_{m^i \in M^i} \max_{m \in M} p(R \text{ and } A \text{ accept } m \mid R \text{ and } A \text{ accept } m^i)$$
 (1)

$$P_{R_i} = \max_{m^i \in M^i, e_r \in E_R} \max_{m \in M} p(A \text{ accepts } m \mid A \text{ accepts } m^i, e_r)$$
 (2)

$$P_{A_i} = \max_{m^i \in M^i, e_a \in E_A} \max_{m \in M} p(R \text{ accepts } m \mid R \text{ accepts } m^i, e_a)$$
 (3)

$$P_{T_0} = \max_{e_t \in E_T} \max_{m \notin M(e_t)} p(R \text{ and } A \text{ accept } m \mid e_t)$$

$$\tag{4}$$

$$P_{\overline{RA}_i} = \max_{m^i \in M^i} \max_{(e_r, e_a) \in E_R \circ E_A} \max_{m \in M} p(T \text{ generates } m \mid T \text{ generates } m^i, e_r, e_a)(5)$$

$$P_{\overline{TA}} = \max_{e_t \in E_T} \max_{e_a \in E_A} \max_{m \in M(e_a) \setminus M(e_t)} p(R \text{ accepts } m \mid e_t, e_a)$$

$$\tag{6}$$

We note that P_{O_i} and P_{T_0} are different from similar attacks given in [3] as we define success of an attacker as successful verification by the receiver and successful acceptance by the arbiter while in [3] only the first condition is required.

Information Theoretic Bounds 2.1

We will use following notations throughout the paper.

$$E_X(m^i) = \{e_x \in E_X : m^i \text{ is available for } e_x\}.$$

$$E_X(e_y) = \{e_x \in E_X : p(e_x, e_y) > 0\}.$$

$$M(e_y) = \{m \in M : m \text{ is available for } e_y\}.$$

Theorem 1 gives the information-theoretic lower bounds on the above 6 types of attack.

Theorem 1. In an A^3 -code against collusion attacks we have

- 1. $P_{O_i} \ge 2^{H(E_R, E_A|M^{i+1}) H(E_R, E_A|M^i)}$.
- 2. $P_{R_i} \ge 2^{H(E_A|M^{i+1},E_R)-H(E_A|M^i,E_R)}$. 3. $P_{A_i} \ge 2^{H(E_R|M^{i+1},E_A)-H(E_R|M^i,E_A)}$. 4. $P_{T_0} \ge 2^{H(E_R,E_A|M,E_T)-H(E_R,E_A|E_T)}$.
- 5. $P_{\overline{RA}} \ge 2^{H(E_T|M^{i+1},E_R,E_A)-H(E_T|M^i,E_R,E_A)}$.
- 6. $P_{TA} \ge 2^{H(E_R|M, E_T, E_A) H(E_R|E_T, E_A)}$.

for
$$i = 0, 1, 2, \cdots$$

2.2 Combinatorial Bounds on Key Spaces

To derive combinatorial bounds we will assume that the probability distribution on $E_T \circ E_R \circ E_A$ is uniform. With this assumption we have the following theorem.

Theorem 2. In an A^3 -code if all six kinds of attacks meet their lower bounds, then

- 1. $|E_T| \ge (\prod_{i=0}^l P_{\overline{RA}_i})^{-1} (\prod_{i=0}^l P_{O_i})^{-1}$. 2. $|E_R| \ge P_{T_0}^{-1} (\prod_{i=0}^l P_{O_i})^{-1} (\prod_{i=0}^l P_{R_i})$.
- 3. $|E_A| \ge P_{T_0}^{-1} (\prod_{i=0}^l P_{O_i})^{-1} P_{\overline{TA}} (\prod_{i=0}^l P_{A_i}).$
- 4. $|E_R \circ E_A| \ge P_{T_0}^{-1} (\prod_{i=0}^l P_{O_i})^{-1}$.

An A^3 -code is called l-optimal if,

- (i) all six types of attacks meet their lower bounds in theorem 1, and
- (ii) all inequalities in theorem 2 are satisfied with equalities.

In the following we give structural properties of l-optimal codes in order to analyse their combinatorial structure.

Corollary 1. In an optimal A³-code against collusion attacks the following properties are satisfied.

- 1. If $E_{\overline{RA}}(m^i) \neq \emptyset$, then $|E_{\overline{RA}}(m^i)|$ is independent of m^i . 2. If $E_T(m^i) \cap E_T(e_r, e_a) \neq \emptyset$, then $|E_T(m^i) \cap E_T(e_r, e_a)|$ is independent of m^i, e_r and e_a .
- 3. If $E_R(m^i) \cap E_R(e_a) \neq \emptyset$, then $|E_R(m^i) \cap E_R(e_a)|$ is independent of m^i and
- 4. If $E_A(m^i) \cap E_A(e_r) \neq \emptyset$, then $|E_A(m^i) \cap E_A(e_r)|$ is independent of m^i and
- 5. If $E_T(m^i) \neq \emptyset$, then $|E_T(m^i)|$ is independent of m^i .
- 6. If $E_R(m^i) \neq \emptyset$, then $|E_R(m^i)|$ is independent of m^i .
- 7. If $E_A(m^i) \neq \emptyset$, then $|E_A(m^i)|$ is independent of m^i .

An A^3 -code is called a Cartesian A^3 -code if for any message m, there is a unique source state s which can be encoded to m.

This means that in a Cartesian code M can be partitioned into $M(s_1)$, $M(s_2), \cdots$ such that messages in $M(s_i)$ only correspond to s_i .

More precisely, for an $s \in S$ and $(e_r, e_a) \in E_R \circ E_A$, define

 $M(s) = \{m : s \text{ can be encoded to } m \text{ by some } e_t \in E_T\}.$

Then using Corollary 1, we have $|M(s)| = \frac{|E_T|}{|E_T(m)|}$ is a constant. So

$$\frac{|M|}{|S|} = |M(s)| \text{ for all } s \in S.$$
 (7)

3 Connection with Combinatorial Designs

In this section we study the combinatorial structure of Cartesian l-optimal A^3 -code against collusion attacks. In particular we give a combinatorial structure of E_T , E_R , E_A and $E_R \circ E_A$.

First we recall some definitions.

Definition 1. A block design is a pair (V, B), where V is a set of v points and B is a family of k-subsets (called blocks) of V.

Definition 2. A block design (V, B) is called α -resolvable if the block set B is partitioned into classes C_1, C_2, \dots, C_n with the property that in each class, every point occurs in exactly α blocks.

We will be interested in α -resolvable designs with the following property: There is an integer l, 0 < l < n, such that property (P1) below is satisfied.

(P1) Any collection of i blocks, from i different classes either intersect in μ_i points or do not intersect, i ($1 \le i \le l+1$).

For each participant we define an incidence structure which defines the mapping given by the participant's keys. For receiver and arbiter, the incidence structure is given by a 0,1 matrix, whose rows are labelled by the participant's keys and columns are labelled by M, and the element in row e and column m is 1 if $e(m) \in S$ and 0, otherwise. For transmitter, it is given by a matrix whose rows are labelled by transmitter's key, and its columns are labelled by S and the element in row e and column s is m if $e_t(s) = m$.

In the following subsections we will study the relationships between combinatorial designs and each participant's incidence structure.

3.1 E_R

Theorem 3. In an l-optimal A^3 -code against collusion attacks, design $(E_R, \{E_R(m) : m \in M\})$ is $\alpha(R)$ -resolvable with property **(P1)**. It has the parameters: $\alpha(R) = \frac{|M|}{|S|} P_{A_0},$ $\mu_i(R) = |E_R|(P_{O_0}P_{O_1} \cdots P_{O_{i-1}})(P_{R_0}P_{R_1} \cdots P_{R_{i-1}})^{-1}, 1 \le i \le l+1.$

$3.2 \quad E_A$

By 7 of Corollary $\mathbb{1}|E_A(m)|$ is a constant. So $(E_A, \{E_A(m) : m \in M\})$ forms a block design.

Theorem 4. In an l-optimal A^3 -code against collusion attacks, design $(E_A, \{E_A(m) : m \in M\})$ is $\alpha(A)$ -resolvable with property **(P1)**. It has the parameters: $\alpha(A) = \frac{|M|}{|S|} P_{R_0},$ $\mu_i(A) = |E_A| (P_{O_0} P_{O_1} \cdots P_{O_{i-1}}) (P_{A_0} P_{A_1} \cdots P_{A_{i-1}})^{-1}, 1 \le i \le l+1.$

3.3 $E_R \circ E_A$

Note that in an optimal code $E_R \circ E_A$ corresponds to an α -resolvable design as well. It has the following properties:

(P2) For any collection of l+1 blocks $B_{j_1}, B_{j_2}, \dots, B_{j_{l+1}}$ from different classes $C_{j_1}, C_{j_2}, \dots, C_{j_{l+1}}$, and any $u \neq j_1, j_2, \dots, j_{l+1}$, there exists a unique block $B_u \in C_u$ such that

$$B_{j_1} \cap \cdots \cap B_{j_{l+1}} \cap B_u = B_{j_1} \cap \cdots \cap B_{j_{l+1}}$$

if $B_{j_1} \cap \cdots \cap B_{j_{l+1}} \neq \emptyset$. Furthermore, for any $B \in \mathcal{C}_u \setminus \{B_u\}, |B_{j_1} \cap \cdots \cap B_{j_{l+1}} \cap B| = 1$

(P3) The point set \mathcal{V} is partitioned into subsets $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_h$ such that for each subset \mathcal{V}_i , $(\mathcal{V}_i, \mathcal{B}'_i)$ is an α -resolvable design with property (P1) and (P2). Here we use \mathcal{B}'_i to denote $\{B_j \cap \mathcal{V}_i \neq \emptyset : B_j \in \mathcal{B}\}$.

From 1 of Corollary 1 we know $|E_{\overline{RA}}(m)|$ is a constant for all $m \in M$. Then $(E_R \circ E_A, \{E_{\overline{RA}}(m) : m \in M\})$ forms a block design. For this block design we have following theorem.

Theorem 5. In an l-optimal A^3 -code against collusion attacks, design $(E_R \circ E_A, \{E_{\overline{RA}}(m) : m \in M\})$ is α -resolvable with properties **(P1)**, **(P2)** and **(P3)**. It has the parameters:

$$\alpha = \frac{|M|}{|S|} P_{O_0},$$

$$\mu_i = |E_R \circ E_A| \prod_{i=0}^{i-1} P_{O_i}, \ 1 \le i \le l+1.$$

3.4 E_T

In order to describe E_T , we recall some definitions to be used later.

Definition 3. A t-design is a block design (V, B) so that any t-subset of V occurs in exactly λ blocks.

Definition 4. A partially balanced t-design is a block design (V, B), where every t-subset of V either occurs in exactly λ blocks or does not occur in any block.

We denote this design by $t-(v,k;\{\lambda,0\})$ -design, where v is the total number of points and k is the size of a block.

Definition 5. A $t - (v, k; \{\lambda, 0\})$ -design (V, B) is a strong partially balanced t-design if it is a $i - (v, k; \{\lambda_i, 0\})$ -design, for any i, 1 < i < t, and also a 1-design.

Definition 6. A $t-(v,k;\{\lambda,0\})$ -design (V,B) is a resolvable partially balanced t-design if the block set can be partitioned into classes $C_1,C_2,\cdots,C_{n'}$ with the property: For each $j(1 \leq j \leq n')$, (V,C_j) is a $t-(v',k;\{\lambda',0\})$ -design.

Definition 7. A strong partially balanced $t - (v, k; \{\lambda, 0\})$ -design (V, B) is resolvable if it is resolvable with classes $C_1, C_2, \dots, C_{n'}$ and property: For each $j(1 \le j \le n')$, (V, C_j) is a strong partially balanced $t - (v', k; \{\lambda', 0\})$ -design.

Now consider design (M, E_T) .

Theorem 6. In an l-optimal A^3 -code against collusion attacks, design (M, E_T) is a $(l+1)-(|M|,|S|;\{\lambda,0\})$ -design which is strong and resolvable, it has parameters as follows:

$$\lambda = \lambda_{l+1} = 1,$$

$$\lambda_i = (P_{O_i} P_{O_{i+1}} \cdots P_{O_l})^{-1} (P_{\overline{RA}_i} P_{\overline{RA}_{i+1}} \cdots P_{\overline{RA}_l})^{-1}, \ 1 \le i \le l,$$

$$\lambda_i' = (P_{\overline{RA}_i} P_{\overline{RA}_{i+1}} \cdots P_{\overline{RA}_l})^{-1}, \ 1 \le i \le l.$$

It is known that in an optimal A-code, the transmitter and the receiver have a partially balanced t-design (see [5]), or an orthogonal array if the code is Cartesian (see [1]). In an optimal Cartesian A^2 -code the transmitter has a partially balanced t-design, while the receiver has an α -resolvable design (see [6]). Our results in this section show that in an optimal Cartesian A^3 -code against collusion attacks, the transmitter has a partially balanced t-design, while the receiver and the arbiter have α -resolvable designs each.

4 Optimal Code from Finite Geometry

In this section we show an example of 1-optimal A^3 -code against collusion attacks.

Let GF(q) be a field of q elements, PG(n,q) be a n-dimensional projective space over GF(q). Every point on PG(n,q) has a homogeneous coordinate $(x_0, x_1, x_2, \cdots, x_n)$, here not all x_i are zero. Two homogeneous coordinates (x_0, x_1, \cdots, x_n) and $(x_0, x_1', \cdots, x_n')$ identify the same point if and only if there is a nonzero constant c such that $(x_0, x_1, \cdots, x_n) = c(x_0', x_1', \cdots, x_n')$. A k-flat H_k $(0 \le k \le n)$ on PG(n,q) is defined as a system of n-k linear equations:

$$\begin{cases} a_{10}x_0 + a_{11}x_1 + \dots + a_{1n}x_n = b_1, \\ a_{20}x_0 + a_{21}x_1 + \dots + a_{2n}x_n = b_2, \\ \dots \\ a_{n-k,0}x_0 + a_{n-k,1}x_1 + \dots + a_{n-k,n}x_n = b_{n-k}, \end{cases}$$

where the rank of coefficient matrix (a_{ij}) is n - k. Thus a 0-flat is a point, a 1-flat is a line, a 2-flat is a plane, and so on.

Now consider PG(4,q) to construct our code. Let each message correspond to a 3-flat in PG(4,q). Fix a line L_S in PG(4,q). Let the points on L_S be regarded as source states. Let the transmitter's encoding rule be a 2-flat e_t not intersecting L_S . Let the receiver's decoding rule and the arbiter's arbitration rule be points e_r and e_a , respectively. The pair (e_t, e_r, e_a) is valid if and only if e_r, e_a are on e_t . A source state s will be encoded to $m = \langle s, e_t \rangle$ which is a 3-flat passing through s and s and s and s are receiver accepts a message s if and only if s in s arbitrates that s should be accepted by the receiver if and only if s is in s. This code is optimal with the parameters:

$$|S| = q + 1,
|M| = q^4 + q^3,
|E_T| = q^6,
|E_R| = q^4 + q^3 + q^2,
|E_A| = q^4 + q^3 + q^2,$$

5 Conclusion

In this paper we introduced a new model for A^3 -codes, obtained information theoretic and combinatorial bounds on security and efficiency parameters of the codes, defined optimal codes and finally derived combinatorial structure of optimal Cartesian codes. Our study of the optimal A^3 -codes is limited to Cartesian codes. Combinatorial structure of optimal A^3 -codes in the general case is an open problem.

References

- E. F. Brickell and D. R. Stinson. Authentication codes with multiple arbiters. In Advances in Cryptology-EUROCRYPT'88, Lecture Notes in Computer Science, volume 330, pages 51-55. Springer-Verlag, Berlin, Heidelberg, New York, 1988. 390, 397
- Y. Desmedt and M. Yung. Arbitrated unconditionally secure authentication can be unconditionally protected against arbiter's attack. In Advances in Cryptology-CRYPTO'90, Lecture Notes in Computer Science, volume 537, pages 177-188.
 Springer-Verlag, Berlin, Heidelberg, New York, 1990. 390
- 3. T. Johansson. Further results on asymmetric authentication schemes. Information and Computation, 151, 1999. 390, 391, 393
- S. Obana and K. Kurosawa. A²-code=affine resolvable + BIBD. In Proc. of ICICS, Lecture Notes in Computer Science, volume 1334, pages 118-129. Springer-Verlag, Berlin, Heidelberg, New York, 1997. 390
- 5. D. Pei. Information-theoretic bounds for authentication codes and block designs. *Journal of Cryptology*, 8:177-188, 1995. 397
- D. Pei, Y. Li, Y. Wang, and R. Safavi-Naini. Characterization of optimal authentication codes with arbitration. In *Proceeding of A CISP '99, Lecture Notes in Computer Science*, volume 1587, pages 303-313. Springer-Verlag, Berlin, Heidelberg, New York, 1999. 390, 397
- G. J. Simmons. Authentication theory/coding theory. In Advances in Cryptology-CRYPTO'84, Lecture Notes in Computer Science, volume 196, pages 411-431.
 Springer-Verlag, Berlin, Heidelberg, New York, 1984. 390
- G. J. Simmons. A cartesian construction for uncondetionally secure authentication codes that permit arbitration. *Journal of Cryptology*, 2:77-104, 1990. 390
- R. Taylor. Near optimal unconditionally secure authentication. In Advances in Cryptology-EUROCRYPT'94, Lecture Notes in Computer Science, volume 950, pages 244-253. Springer-Verlag, Berlin, Heidelberg, New York, 1994. 390
- Y. Frankel Y. Desmedt and M. Yung. Multi-receiver/multi-sender network securety: efficient authenticated multicast/feedback. In *IEEE Infocom*, pages 2045-2054, 1992. 390

Broadcast Authentication in Group Communication

Rei Safavi-Naini¹ and Huaxiong Wang²

¹ School of IT and CS, University of Wollongong, Australia rei@uow.edu.au

Abstract. Traditional point-to-point message authentication systems have been extensively studied in the literature. In this paper we consider authentication for group communication. The basic primitive is a multireceiver authentication system with dynamic sender (DMRA-code). In a DMRA-code any member of a group can broadcast an authenticated message such that all other group members can individually verify its authenticity. In this paper first we give a new and flexible 'synthesis' construction for DMRA-codes by combining an authentication code (Acode) and a key distribution pattern. Next we extend DMRA-codes to tDMRA-codes in which t senders are allowed. We give two constructions for tDMRA-codes, one algebraic and one by 'synthesis' of an A-code and a perfect hash family. To demonstrate the usefulness of DMRA systems, we modify a secure dynamic conference key distribution system to construct a key-efficient secure dynamic conference system that provides secrecy and authenticity for communication among conferencess. The system is key-efficient because the key requirement is essentially the same as the original conference key distribution system and so authentication is effectively obtained without any extra cost. We show universality of 'synthesis' constructions for unconditional and computational security model that suggests direct application of our results to real-life multicasting scenarios in computer networks. We discuss possible extensions to this work.

1 Introduction

Collaborative and multi-user applications, such as teleconferences and electronic commerce applications, require secure communication among members of a group. Compared to providing confidentiality, ensuring integrity and authenticity of information is much more difficult as in the latter subgroups of participants can participate in a coordinated attack against other group members, while in the former they are passive. It is also worth emphasizing that the two goals of confidentiality and authenticity in group communications are independent and achieving one goal does not give assurance about the other goal.

We consider the following scenario. There is a group of users and a Trusted Authority (TA). During the initialization of the system, TA generates keys for

Department of Computer Science, National University of Singapore, Singapore wanghx@comp.nus.edu.sg

all participants and securely delivers the keys to them. After this stage, each user can broadcast a message which is verifiable for its origin and integrity by every other user, individually. We assume that users are not all trusted and collude to construct a fraudulent message which they will attribute to another user. We assume security is unconditional and does not rely on any computational assumption.

The obvious method of providing protection in the above scenario is to use a conventional point-to-point authentication system and give a shared key to each pair of users. Now to construct an authenticated message, a user will construct the authenticator for every other user, concatenate all the authenticators and append it to the message. Each receiver can individually verify authenticity of the message by examining the authenticator constructed using his shared key. Two immediate drawbacks of this system are, 1) it requires a very large key storage, and 2) it produces a very long tag for a message resulting in high communication cost. A more serious problem is that the construction does not provide any security. This will become clear later in the section.

Multireceiver authentication systems (MRA-codes) [8] can be seen as the first attempt at providing authentication in group communication. However in this model the sender is fixed. In [18] this limitation is removed and the sender can be any group member. The extended model is called MRA-code with dynamic sender, or DMRA-codes for short. DMRA-codes capture the essential aspect of authentication in group communication but the model allows only one sender while in many applications such as dynamic conference key distributions, group members interact with each other and more than one sender exists. Moreover, the only known non-trivial construction [18] is very inflexible and for large size groups, or large size sources, results in very inefficient constructions with many key bits and long authenticators. In summary, although DMRA-codes do provide a promising starting point for authentication in group communication, for practical applications more general, flexible and efficient models and constructions are required.

Our goal is to have solutions that are efficient both in terms of storage and communication cost. To achieve our goal we assume the following.

- The largest size of collusion set is w.
- There are at most t transmitters (senders).

These are both reasonable assumptions. The first assumption effectively bounds the power of attackers, and the second one is similar to the degree of spoofing in a conventional authentication system but is more complex to protect against as the t messages are from different originators and so a new type of attack, that is changing the originator of a message, is introduced.

This new attack points to the fact that in a DMRA-code with t dynamic senders an authenticated message must carry some information about its origin. The attack works because in general we allow the same message to be sent by more than one sender. This is a realistic requirement in many applications such as a

voting system with many participants and only few possible message, for example only a 'yes' or 'no' answer. In this attack, that we call directional attack, an intruder firstly changes the origin information of a message that is already sent by P_j , and then resends and attributes it to P_i . This could give P_i a higher success chance if P_i and P_j share some key information which is used for generation of authenticators. This observation immediately rules out direct application of schemes that establish a common key among two or more users, including the scheme described above, the construction based on symmetric polynomial [18], or application of KDP and its more general form (i,j)-cover-free family [23] for key distribution. It is worth noting that all such constructions can be immediately used to provide confidentiality in group communication, but exactly because they result in a shared key among two or more participants they cannot be used in group authentication systems.

To include information about the origin in a broadcasted message, a simple approach would be to attach identity information to the message and then authenticate the result. Although this added information could protect against directional attack but will effectively increase the size of the of message space, which in the case of a small source and a large number of participant such as the voting system mentioned above, is not acceptable. We will show that this information is theoretically redundant and can be removed in an optimal system. In the rest of this paper we assume that identity information is appended to the authenticated message. The contributions of this paper can be summarized as follows.

- 1. We formalize the model of DMRA-codes to allows more than one sender. This generalizes the model given in [18,20].
- 2. We start with DMRA-code with a single sender. We propose a new, general 'synthesis' construction for DMRA-codes by combining a key distribution pattern (KDP) [16] and an A-code, such that the protection of the resulting system can be determined by the protection of the underlying A-code and parameters of the KDP. The construction is especially attractive as it reduces construction of a DMRA-code to the construction of suitable KDPs and A-codes, and so allows a direct application of the previously known results in these latter two areas to the construction of DMRA-codes.
- 3. We then consider a DMRA-code with t senders, tDMRA-code for short, and give two new constructions for such systems. The first construction is algebraic and uses polynomials in two variables. The second construction is a 'synthesis' construction which is combinatorial in nature and combines a perfect hash family and an arbitrary A-code to obtain a tDMRA-code.
- 4. To show the applicability of our results, we give an interesting application of DMRA-codes by constructing a secure dynamic conference system that also provides the authenticity. Our construction is built on an optimal dynamic conference key distribution system proposed in [6], and for large group sizes, as long as the conference size is relatively small, effectively adds authentication without any extra cost (extra key bits).

5. Although our analysis is for Cartesian A-codes in the context of unconditional security our main constructions are universal. That is they can also be used for A-codes with secrecy and MACs (Message Authentication Codes), resulting in tDMRA-codes in which protection is determined by the security property of the underlying primitive A-system (A-code with or without secrecy, and MAC) and parameters of a combinatorial structure, a KDP or a perfect hash function family.

The paper is organized as follows. In section 2 we give the model and definitions. In section 3 we briefly review previous results for single sender case and describe a new, flexible construction from key distribution patterns and A-codes. In section 4 we consider systems with multiple senders and present two constructions. In section 5 we propose a secure dynamic conference system and in section 6 discuss computationally secure systems. Section 7 concludes the paper.

2 The Model

A (w, n) MRA-code [8] is an interesting extension of the classical authentication systems (A-systems for short) where a *fixed* sender can authenticate a *single* message for a group of n receivers such that collusions of up to w receivers cannot construct a fraudulent codeword which is accepted by another receiver. Bounds and construction for MRA-codes are given in [11,13,18].

An extension of the MRA-code model is when the sender is not fixed and can be any member of the group. We call the system MRA-code with dynamic sender. Allowing the sender to be dynamic introduces the notion of authenticating with respect to a particular originator. That is, to verify authenticity of a received message a receiver must first assume an originator for the message and then verify authenticity of the message with respect to that particular originator. Thus a broadcast message in general carries information about its origin, together with its real content.

In the model of MRA-code with dynamic senders, there are n users $\mathcal{P} = \{P_1, \ldots, P_n\}$, who want to communicate over a broadcast channel. The channel is subject to spoofing attack: that is a codeword can be inserted into the channel or, a transmitted codeword can be substituted with a fraudulent one. An attack is directed towards a channel, consisting of a pair of users $\{P_i, P_j\}$, P_i as the sender and P_j as the receiver. A spoofer might be an outsider, or a coalition of w insiders. The aim of the spoofer(s) is to construct a codeword that P_j accepts as being sent from P_i .

We assume there is a Trusted Authority(TA) who generates and distributes secret keys for each users. The TA is only active during key distribution phase. The system consists of three phases.

1. Key Distribution: The TA randomly chooses a key $e \in E$ and applies a key distribution algorithm

$$\tau: E \longrightarrow E_1 \times \cdots \times E_n, \quad \tau(e) = (e_1, \dots, e_n)$$

to generate a key e_i for user P_i , $1 \le i \le n$, and secretly sends e_i to P_i .

2. Broadcast: A user P_i constructs an authenticated message and broadcasts it. For this, P_i uses his/her own authentication algorithm,

$$A_i: S \times E_i \longrightarrow M_i, \quad A_i(s, e_i) = m_i,$$

where E_i and M_i are the set of keys and authenticated codeword for P_i . The codeword sent by P_i for a source state $s \in S$ is $(i, A_i(s, e_i)) = (i, m_i)$.

3. Verification: A user P_j , j, $1 \le j \le n$, uses his/her verification algorithm \mathcal{V}_{ji} to accept or reject the received codeword. That is, the key e_j determines a set of verification algorithms $\{\mathcal{V}_{ji}; 1 \le i \le n, j \ne i\}$ with

$$\mathcal{V}_{ii}: M_i \times E_i \longrightarrow \{0,1\},$$

such that $V_{ji}(m_i, e_j) = 1$ if P_j accepts m_i as an authenticated codeword sent from P_i and $V_{ji}(m_i, e_j) = 0$ otherwise.

We assume that after a key distribution phase, there are at most t users who broadcast their authenticated messages and the messages all come from a set S of source states. For simplicity, we also assume that each sender may only broadcast a single message. We will adopt the Kerckhoff's principle and assume details of the system, except the actual keys, are public. We call the system a (w,n) tDMRA-code, where w is the largest size of the collusion members, and represent it by $C = (S, E, \{M_i, E_i\}_{1 \le i \le n})$, or in Cartesian authentication system, by $C = (S, E, \{A_i, E_i\}_{1 \le i \le n})$.

To assess the security, we define the probability of success in various attacks. Let B and A be two subsets of $\{1,\ldots,n\}$ with $|B|=\beta\leq t$ and $|A|=\alpha\leq w$. Without loss of generality, let $B=\{k_1,\ldots,k_\beta\}$ and $A=\{\ell_1,\ldots,\ell_\alpha\}$, and denote $P_B=\{P_{k_1},\ldots,P_{k_\beta}\}$ and $P_A=\{P_{\ell_1},\ldots,P_{\ell_\alpha}\}$. Assume that after seeing the authenticated messages $(s_{k_1},a_{k_1}),\ldots,(s_{k_\beta},a_{k_\beta})$ broadcast by $P_{k_1},\ldots,P_{k_\beta}$, respectively $(s_{k_1},\ldots,s_{k_\beta})$ are not necessary distinct, P_A want to generate a message (s_i,a_i) such that it will be accepted by P_j as an authenticated message broadcast by P_i . We further assume that $i,j\not\in A$.

Let $P[P_A, P_B, P_i, P_j]$ denote the probability of success for malicious users P_A in constructing a fraudulent message that P_j accepts as authentic and broadcast by P_i , after the broadcast messages from P_B are seen. We assume the malicious users use their optimal strategy and want to maximize their chance of success. They can choose the message and the channel, that is P_i , P_j , to achieve this goal.

It is easy to see that if $A \subset A'$, then $P[P_A, P_B, P_i, P_j] \leq P[P_{A'}, P_B, P_i, P_j]$. Thus, without loss of generality, we assume that |B| = w. For each $0 \leq k \leq t$, we define

$$P_{D_k} = \max_{A,B,i,j} P[P_A, P_B, P_i, P_j]$$

where the maximum is taken over all possible A, B, i, j such that |A| = w, |B| = k and $i, j \in A$. We then define the overall probability of deception, denoted by P_D , as

$$P_D = \max\{P_{D_0}, P_{D_1}, \dots, P_{D_t}\}.$$

3 DMRA-Codes with a Single Sender

We start with the simplest (w, n) tDMRA-code in which t = 1 and simply call it (w, n) DMRA-code. This is exactly the same model as MRA-code with dynamic sender introduced in [18]. In section 3.1 we briefly review combinatorial lower bounds on the key size for each user, and also the length of the authenticator.

3.1 Bounds and an Optimal Construction

Efficiency of a (w, n) DMRA-code $C = (S, E, \{M_i, E_i\}_{1 \le i \le n})$ can be measured in terms of the size of each user's key space, $|E_i|$ and the length of the authenticated message, $|M_i|$. We do not really need to consider the size of key for TA, |E|, as after the key distribution phase TA does not need to remember the key and so can erase his key. The following lower bounds can be used to determine the best performance of a DMRA-code.

Theorem 1 ([20]). In a (w, n) DMRA-code $C = (S, E, \{M_i, E_i\}_{1 \le i \le n})$ with $P_D \leq 1/q$ and uniform probability distribution on the source S, we have:

(i)
$$|E_i| \ge q^{2(w+1)}$$
, for each $i \in \{1, 2, ..., n\}$,
(ii) $|M_i| \ge q^{w+1}|S|$, for each $i \in \{1, 2, ..., n\}$

(ii)
$$|M_i| \ge q^{w+1}|S|$$
, for each $i \in \{1, 2, ..., n\}$

The bounds in Theorem 1 are tight. In [20], we give a construction to meet the bounds with equality.

3.2 A General Construction

In the following we show a general construction for (w, n) DMRA-codes by combining a key distribution pattern and an A-code such that the security of the resulting system is determined by the security of the underlying A-code and the parameters of the key distribution pattern. The importance of this construction is that it provides a much higher degree of flexibility in the design of DMRAcodes and results in constructions that are practical.

Key distribution patterns (KDP) [16] are explicitly or implicitly used by numerous authors to construct key distribution systems [9,12,14,17,22,23]).

Let $X = \{x_1, x_2, \dots, x_v\}$ be a set, and $\mathcal{B} = \{B_1, B_2, \dots, B_n\}$ be a family of subsets of X. The set system (X, \mathcal{B}) is called a (n, v, w) key distribution pattern (or KDP(n, v, w) for short) if

$$|(B_i \cap B_j) \setminus (\cup_{s=1}^w B_{\ell_s})| \ge 1$$

for any w + 2 subset $\{i, j, \ell_1, ..., \ell_w\}$ of $\{1, 2, ..., n\}$.

Assume there are n users P_1, \ldots, P_n . Let (X, \mathcal{B}) be a KDP(n, v, w) and (S, A_0, E_0) be a Cartesian authentication code such that the probability of deception (impersonation and substitution attacks) is bounded by 1/q. Associate B_i to P_i , $1 \le i \le n$. Both (X, \mathcal{B}) and (S, A_0, E_0) are public.

- Key distribution: For each 1 ≤ j ≤ v, TA randomly chooses an authentication key e_j ∈ E₀ and gives e_j to user P_i if x_i ∈ B_j. Thus, user P_i receives a |B_i|-tuple, (e_{i1},..., e_{i|B_i|}) ∈ E₀^{|B_i|}, as his/her secret authentication key.
 Broadcast: When P_i wants to construct an authenticated message for a source
- 2. Broadcast: When P_i wants to construct an authenticated message for a source state $s \in S$, he computes $|B_i|$ partial authenticators $e_{i_t}(s)$, $1 \le t \le |B_i|$, and broadcasts $(s, e_{i_1}(s), \ldots, e_{i_{|B_i|}}(s))$ together with his identity i.
- 3. Verification: A user can verify authenticity and origin of the broadcast message in the following way: P_j uses the origin information, i, to determine the set $E_{ij} = \{e_{i_t}\}_{1 \leq t \leq |B_i|} \cap \{e_{j_k}\}_{1 \leq k \leq |B_j|}$ and accepts $(s, e_{i_1}(s), \ldots, e_{i_{|B_i|}}(s))$ as authentic and sent from P_i if for all $e \in E_{ij}$, e(s) is the same as the corresponding component in $(e_{i_1}(s), \ldots, e_{i_{|B_i|}}(s))$.

Theorem 2. Let the deception probability of the underlying A-code (S, A_0, E_0) be bounded by 1/q. Then the above construction results in a (w, n) DMRA-code $C = (S, E, \{A_i, E_i\}_{1 \le i \le n})$ with $P_D \le 1/q$. The code has the following parameters

$$|E| = |E_0|^v$$
, $|E_i| = |E_0|^{|B_i|}$ and $|A_i| = |A_0|^{|B_i|}$.

The construction also works for general A-codes in which case the broadcast codeword by P_i is $(m_{i_1} \cdots m_{i_{|B_i|}})$ where $m_{i_j} = e_{i_j}(s)$.

The main advantages of the construction is its flexibility in the choice of parameters for different applications.

4 DMRA-Codes with Multiple Senders

In this section we consider tDMRA-codes, with $t \geq 2$. In designing a (w,n) tDMRA-code with $t \geq 2$, it is important to note that if the protection for a channel between two participants P_i and P_j is provided by a symmetric key system, then a message sent by P_i can be later resent and claimed to have come from P_j . In this case P_i will accept the message as authentic from P_j and the success chance of the intruder is 1. To avoid this directional attack, P_i and P_j must have different keys.

Trivial Construction 1. An obvious method of constructing a tDMRA-code is to use t copies of a DMRA-code with t independent keys. That is, in the key generation phase the TA chooses t independent keys, e^1 , e^2 , $\cdots e^t$, for a DMRA-code and gives the user P_i , a t tuple, $(e_i^1, e_i^2, \cdots e_i^t)$. A user P_i will use key e_i^ℓ to authenticate (generate or verify) the ℓ^{th} message. In this case the size of the key for each participant is t times that of a DMRA-code, which for efficient DMRA-codes and small t could be reasonably low. The length of the tag for each message is the same as the original DMRA-code. However the system is unacceptable as it requires each user to carefully keep track of all the communicated messages and use the correct key for each particular message. If a message is missed, all future communications will be lost.

Trivial Construction 2. A second immediate solution will be to use a (w, n-1) MRA-code. In this case each user receives the key information for sending one

message, and the key information for verifying n-1 messages. The result is a (w,t) tDMRA-code with t=n. The length of tag in this case is the same as the MRA-code but the key storage is at least a linear function of n. This means that the key storage will be prohibitively large for large groups.

4.1 A Polynomial Construction for tDMRA-Codes

The first non-trivial construction uses polynomials in two variables over finite fields. Let S=GF(q) be the set of source states. We construct a (w,n) tDMRA-code as follows.

- 1. Key Distribution: The TA randomly chooses two polynomials F(x,y) and G(x,y) with two variables x and y of degree at most w and w+t, respectively. Then he chooses n distinct numbers $a_i \in GF(q)$ and n distinct numbers $b_i \in GF(q)$, where (a_i, b_i) is P_i 's identity information and is public. For each $i, 1 \le i \le n$, the TA privately sends two pairs of polynomials $(F(x, a_i), G(x, a_i)), (F(b_i, y), G(b_i, y))$ to P_i . This constitutes the secret information of P_i .
- 2. Broadcast: If P_i wants to authenticate a message $s_i \in GF(q)$, P_i calculates the polynomial $U_i(x) = F(x, a_i) + s_i G(x, a_i)$ and broadcasts $(s_i, U_i(x))$ and his identity a_i to other users.
- 3. Verification: P_j can verify the authenticity of $(s_i, U_i(x))$ by calculating the polynomial $V_j(y) = F(b_j, y) + s_i G(b_j, y)$ and accepting $(s_i, U_i(x))$ as authentic and being sent from P_i if $V_j(a_i) = U_i(b_j)$.

Theorem 3. The above construction results in a (w,n) tDMRA-code $C = (S, E, \{A_i, E_i\}_{1 \leq i \leq n})$, in which the probability of success for a collusion of up to w users in performing impersonation or substitution attacks on any other pair of users is at most 1/q. The construction has the following parameters

$$|E| = q^{2(w+1)(t+w+1)}, |E_i| = q^{4w+2t+4} \text{ and } |A_i| = q^{w+1}, 1 \le i \le n.$$

Comparing this construction with trivial construction 2, shows marked improvement of efficiency. In particular, let n independent copies of the optimal MRA-codes based on (polynomial) scheme [8] be used. It will result in a (w,n) tDMRA-code $C = (S, E, \{M_i, E_i\}_{1 \le i \le n})$ with parameters $|E| = q^{(\frac{(w+1)w}{2} + n)}$ and $|E_i| = q^{2(n-1+w)}$ and $|A_i| = q^{w+1}$ and so the size of the key space for the TA and users are $O(n \log q)$. Thus, when n is much larger than t, the key storage for the TA and users can be significantly reduced.

4.2 A General Construction from Perfect Hash Families

In this section, we present a general approach to the construction of tDMRA-codes by combining a general A-code and a perfect hash family.

A (n, m, w)-perfect hash family (PHF for short) is a set of functions \mathcal{F} such that

$$f: \{1, \ldots, n\} \longrightarrow \{1, \ldots, m\}$$

for each $f \in \mathcal{F}$, and for any $X \subseteq \{1, \ldots, n\}$ such that |X| = w, there exists at least a function f^X in \mathcal{F} such that f^X is an injection on X, *i.e.* the restriction of f^X on X is one-to-one. For a subset X, if the restriction of a function f on X is one-to-one, then we call f perfect on X. We will use the notation PHF(N; n, m, w) for a (n, m, w) perfect hash family with $|\mathcal{F}| = N$. PHFs have been applied to cryptographic applications such as broadcast encryption systems [10], secret sharing schemes [4], and threshold cryptography [3].

Let $C_0 = (S, E_0, A_0)$ be a Cartesian A-code (code without secrecy) with deception $P_D \leq \epsilon$, and assume $\mathcal{F} = \{f_1, \ldots, f_n\}$ be a $PHF(N; n, n_0, w + t + 2)$ from $\{1, \ldots, n\}$ to $\{1, \ldots, n_0\}$. We construct a (w, n) tDMRA-code $C = (S, E, \{A_j, E_j\}_{1 \leq j \leq n})$ as follows.

1. Key Distribution: The TA randomly chooses N $n_0 \times n_0$ matrices

$$G^1 = (g_{u,v}^1)_{1 \le u \le n_0, 1 \le v \le n_0}, \dots, G^N = (g_{u,v}^N)_{1 \le u \le n_0, 1 \le v \le n_0}$$

with entries $g_{u,v}^{\ell} \in E_0$, for all $1 \leq \ell \leq N$ and $1 \leq u,v \leq n_0$. For each $1 \leq i \leq n$, the TA generates the key e_i of P_i by

and secretly sends e_i to P_i . That is, the secret key of P_i consists of the $f_{\ell}(i)$ th column and the $f_{\ell}(i)$ th row of matrix G^{ℓ} , for all $1 \leq \ell \leq N$.

2. Broadcast: if P_i wants to authenticate a message $s_i \in S$, P_i generate his authenticator a_i for s_i by

$$a_{i} = \begin{pmatrix} g_{1,f_{1}(i)}(s_{i}) \\ \vdots \\ g_{n_{0},f_{1}(i)}(s_{i}) \end{pmatrix}, \dots, \begin{bmatrix} g_{1,f_{N}(i)}(s_{i}) \\ \vdots \\ g_{n_{0},f_{N}(i)}(s_{i}) \end{bmatrix} \end{pmatrix}$$

and broadcasts (s_i, a_i) , together with his identity i, to all other members. That is, P_i uses the *column key* to generates his authenticator.

3. Verification: P_i uses his row key

$$([g_{f_1(j),1},\cdots,g_{f_1(j),n_0}],\cdots,[g_{f_N(j),1},\cdots,g_{f_N(j),n_0}])$$

to verify the authenticity of the broadcast message (s_i, a_i) in the following way. Since for each $1 \leq \ell \leq N$, the $f_{\ell}(j)$ th row and $f_{\ell}(i)$ th column of matrix G^{ℓ} has a common entry $g_{f_{\ell}(j),f_{\ell}(i)}$, which P_j can use to verify if $g_{f_{\ell}(j),f_{\ell}(i)}(s_i)$ is the correct partial authenticator in a_i .

Theorem 4. Suppose there exists a Cartesian A-code $C_0 = (S, E_0, A_0)$ with deception probability $P_D \leq \epsilon$, and a PHF(N; n, $n_0, t + w + 2$). Then the above construction results in a (w, n) tDMRA-code $C = (S, E, \{A_j E_j\}_{1 \leq j \leq n})$ with deception probability $P_D^* \leq \epsilon$. The various parameters satisfy

$$|E| = |E_0|^{Nn_0^2}$$
, $|E_j| = |E_0|^{(2n_0 - 1)N}$ and $|A_j| = |A_0|^{n_0 N}$, $1 \le j \le n$

For a given set of parameters, w, t and n, and a given A-code the efficiency of the scheme is completely determined by N, the size of perfect hash family \mathcal{F} . Let N(n,m,w) denote the minimum value of N such that a PHF(N;n,m,w) exists. Thus we will be interested in perfect hash families with small N(n,m,w) for given n,m and w. In particular, we are interested in the behavior of N(n,m,w) as a function of n, when m and w are fixed. It is proved in [15] that for fixed m and w, N(n,m,w) is $\Theta(\log n)$, however, the proof is non-constructive and PHF that achieve this asymptotic bound are believed to be difficult to construct. $(f(n) = \Theta(\log n))$ means that there exist constants c_1 , c_2 and n_0 such that for $n > n_0$, $c_1 \log n \le f(n) \le c_2 \log n$.) In [1,5] some constructions with reasonable asymptotic performance are given. For example, for fixed m and w, N is a polynomial function of $\log n$. Various other bounds on N(n,m,w) can be found in [15,1,7,3].

5 A Secure Dynamic Conference System

To show the usefulness of group authentication systems we will use DMRA-codes to construct a secure dynamic conference system that also provides the authenticity.

A Key Distribution Systems (KDS) is one of the main primitives for distributing keys in network and group communication [22]. In a KDS, the collection of all subsets of n users is divided into privileged subsets and forbidden subsets. To each privileged subset, G, of users a secret key, k_G , is attached. k_G is computable by each member of G and collusion of members of a forbidden set F, disjoint from G, cannot learn anything about k_G . A TA generates and distributes secret key information to all users. If privileged sets are t-subsets of \mathcal{P} , and forbidden sets are all w-subsets of \mathcal{P} , we use the notation (t,w) KDS. Blundo, De Santis, Herzberg, Kutten, Vaccaro and Yung [6] (BDHKVY for short) proposed a (c,w)-KDS in which each user has to store $\binom{c+w-1}{c-1}\lceil\log q\rceil$ bits of key while the TA has to store $\binom{c+w}{c}\lceil\log q\rceil$ bits of key, they also proved that these are the minimum possible storage requirements for both the TA and each users.

Given a (c, w)-KDS, we can easily construct a broadcast encryption system [10] in the following way. Assume that the TA wants to send a message $s \in GF(q)$ (e.g. a session key) to a group of users P_{j_1}, \ldots, P_{j_c} , or one of the users, P_{j_1} , wants to send s to other users in $\{P_{j_2}, \ldots, P_{j_c}\}$. The TA, or P_{j_1} , encrypts s as $b = s + k_{j_1, \ldots, j_c}$ and broadcasts b. Then any user in $\{P_{j_1}, \ldots, P_{j_c}\}$ can decrypt b to obtain s, by using $s = b - k_{j_1, \ldots, j_c}$, and any group of at most w users that are disjoint from $\{P_{j_1}, \ldots, P_{j_c}\}$ have no information about s.

However the communication is not authenticated. That is, the origin of a message is not known and hence there is no accountability in the system. In the following we show how to add authenticity to this system without having more key bits.

- 1. Key Distribution: Assume that there is a BDHKVY (c, w)-KDS, where the TA has randomly chosen a symmetric polynomial $P(x_1, \ldots, x_c)$ in c variables and of degree at most w, and privately transmitted the secret information $P(i, x_2, \ldots, x_c)$ to each user P_i . The field GF(q) is chosen such that $q \ge \max\{|S|, n+2\binom{n}{c}+c-2\}$, where S is the set of source states. To each group of users, $\{P_{j_1}, \ldots, P_{j_c}\}$, of size c we associate a number N_{j_1, \ldots, j_c} such that $n < N_{j_1, \ldots, j_c} \le 2\binom{n}{c}$ and such that if $\{P_{j_1}, \ldots, P_{j_c}\} \ne \{P_{j'_1}, \ldots, P_{j'_c}\}$ then $|N_{j_1, \ldots, j_c} N_{j'_1, \ldots, j'_c}| \ge 2$. The numbers N_{j_1, \ldots, j_c} will serve as identity information for conferences and are made public.
- 2. Broadcast: Assume that P_{j_1} wants to encrypt a message $s \in S$ and broadcast it such that each user in $\{P_{j_2}, \ldots, P_{j_c}\}$ can decrypt the message and individually verify the authenticity and the origin of the message.
 - (a) P_{j_1} constructs two polynomials, of degree at most w,

$$F_{j_1}(x_2) = f_{j_1}(x_2, N_{j_1, \dots, j_c}, \dots, N_{j_1, \dots, j_c} + c - 2)$$

$$G_{j_1}(x_2) = f_{j_1}(x_2, N_{j_1, \dots, j_c} + 1, \dots, N_{j_1, \dots, j_t} + c - 1).$$

 P_{j_1} then encrypts s with the (conference) key $k_{j_1,...,j_c}$ to obtain $b=s+k_{j_1,...,j_c}$.

- (b) P_{j_1} computes the polynomial $A_{j_1}(x_2) = F_{j_1}(x_2) + bG_{j_1}(x_2)$ of degree at most w, and broadcasts $(b, j_1, A_{j_1}(x_2))$.
- 3. Decryption and verification: Each user P_{j_i} in $\{P_{j_2}, \ldots, P_{j_c}\}$ can decrypt and verify the authenticity of the message broadcast by P_{j_1} : in the same manner as (2.1) and (2.2), P_{j_i} can calculate $A_{j_i}(x_2)$. Then, P_{j_i} verifies if $A_{j_i}(j_1) = A_{j_1}(j_i)$ holds and if true, accepts the broadcast codeword as authentic from P_{j_1} . Finally P_{j_i} decrypts b by $s = b k_{j_1, \ldots, j_c}$ to get s.

Theorem 5. For c > 2, the above construction provides secrecy and authenticity for (c, w)-KDS for dynamic conferences.

We note that compared with the broadcast encryption scheme based on the BDHKVY KDS, the key storage of the above scheme need not increase, if $|S| \ge (n+2\binom{n}{c}+c-2)$.

6 Computationally Secure tDMRA-Codes

Security in tDMRA-codes so far has been unconditionally secure model. Although unconditionally secure schemes offer the highest possible security but their key requirement is usually very large and so such systems are usually impractical. In practice, data integrity is obtained by using MACs (message authentication codes) and signature schemes. MACs can be seen as the computationally secure version of A-codes. Numerous constructions for MACs exist.

MACs can be constructed from block cipher systems (for example DES) in CBC mode, or using cryptographic hash functions like MD5 and SHA-1. MACs with provable security can be obtained through Wegman-Carter construction[24].

A very important aspect of 'synthesis' constructions for MRA, DMRA and tDMRA-code is that they work with MACs too. Each 'synthesis' construction essentially combines an A-code with a combinatorial structure: a cover-free family, a KDP or a PHF, respectively. By replacing the A-code with a MAC, a system (MRA, DMRA and tDMRA-code) with computational security is obtained such that the security can be directly related to the security of the underlying MAC and parameters of the combinatorial structure.

This universality of 'synthesis' constructions is especially important because combinatorial techniques, such as constructing systems for large groups that is obtained through recursive constructions using PHFs, can also be imported to computationally secure model.

7 Conclusions

In this paper, we studied broadcast authentication. DMRA-codes are the basic primitive to allow one authenticated message be sent to the group. We gave a flexible construction which combines two smaller structures, an A-code and a KDP, for these systems. Although the constructions assume only one codeword sent by a sender, but it is not difficult to extend them to multiple messages from the sender. When multiple messages are from different senders, a new type of attack must be considered. The aim of the attack is to tamper with the origin information in a broadcast message. Protection against this attack implies that the keys used by the two users must produce uncorrelated tags and so key distribution systems that establish common key among participant cannot be directly used for key distribution in group authentication systems. We gave two constructions for tDMRA-codes, one algebraic and one by a 'synthesis' method. 'Synthesis' constructions are especially interesting as they are universally applicable with A-codes, with and without secrecy, and MACs.

A DMRA-code is a powerful tool for securing group communications. We showed a construction for secure dynamic conference systems which provides confidentiality and authenticity for communicated messages.

The question of optimality of tDMRA-code is only answered when t=1. Deriving information theoretic and combinatorial bounds for general tDMRA-codes, and constructing optimal tDMRA- systems are interesting open problems.

References

- M. Atici, S.S. Magliveras, D. R. Stinson and W.D. Wei, Some Recursive Constructions for Perfect Hash Families. *Journal of Combinatorial Designs* 4(1996), 353-363. 408
- 2. M. Bellare, R. Canetti and H. Krawczyk, Key hash functions for message authentication, in *Advance in Cryptology–Crypto '96*, LNCS, **1109**(1996), 1-15.

- S. R. Blackburn, Combinatorics and Threshold Cryptology, in Combinatorial Designs and their Applications, Chapman & Hall/CRC Res. Notes Math. Vol. 403(1997), 49-70. 407, 408
- S. R. Blackburn, M. Burmester, Y. Desmedt and P. R. Wild, "Efficient multiplicative sharing schemes," in Advance in Cryptology-Eurocrypt '96, LNCS, 1070(1996), 107-118.
- S.R. Blackburn and P.R. Wild, Optimal linear perfect hash families, J. Comb. Theory - Series A, 83(1998), 233-250. 408
- C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, Perfectly secure key distribution for dynamic conferences. *Lecture Notes in Computer Science* 740(1993), 471-486 (Advances in Cryptology CRYPTO'92) 401, 408
- Z. J. Czech, G. Havas and B. S. Majewski, Perfect Hasing, Theoretical Computer Science 182(1997), 1-143. 408
- Y. Desmedt, Y. Frankel and M. Yung, Multi-receiver/Multi-sender network security: efficient authenticated multicast/feedback, *IEEE Infocom'92*, (1992) 2045-2054. 400, 402, 406
- M. Dyer, T. Fenner, A. Frieze and A. Thomason, On key storage in secure Networks. *Journal of Cryptology* 8(1995), 189-200. 404
- A. Fiat and M. Naor, Broadcast Encryption. In "Advances in Cryptology Crypto '93", Lecture Notes in Computer Science 773 (1994), 480-491. 407, 408
- H. Fujii, W. Kachen and K. Kurosawa, Combinatorial bounds and design of broadcast authentication, *IEICE Trans.*, VolE79-A, No. 4(1996)502-506.
- L. Gong and D. J. Wheeler, A matrix key-distribution scheme. J. Cryptology, Vol.2(1990), 51-59.
- 13. K. Kurosawa and S. Obana, Characterization of (k, n) multi-receiver authentication, Information Security and Privacy, ACISP'97, Lecture Notes in Comput. Sci. 1270,(1997) 204-215. 402
- T. Matsumoto, Incidence structures for key sharing, Lecture Notes in Computer Science 917(1995), 242-253(Advances in Cryptology-Asiacrypt '94).
- 15. K. Mehlhorn, Data Structures and Algorithms, Vol. 1, Springer-Verlag, 1984. 408
- C. J. Mitchell and F. C. Piper, Key storage in secure networks. Discrete Applied Mathematics 21(1988), 215-228. 401, 404
- C. M. O'Keefe, Key distribution patterns using Minkowski planes. Designs, Codes and Cryptography 5(1995) 261-267. 404
- R. Safavi-Naini and H. Wang, New results on multi-receiver authentication codes, Advances in Cryptology – Eurocrypt '98, Lecture Notes in Computer Science, 1438(1998), 527-541. 400, 401, 402, 404
- R. Safavi-Naini and H. Wang, Bounds and constructions for multireceiver authentication codes, Advances in Cryptology Asiacrypt '98, Lecture Notes in Computer Science, 242-256.
- R. Safavi-Naini and H. Wang, Multireceiver authentication codes: models, bounds, constructions and extensions, *Information and Computation* 151(1999), 148-172 401, 404
- G. J. Simmons, A survey of information authentication, in Contemporary Cryptology, The Science of Information Integrity, G.J. Simmons, ed., IEEE Press, (1992), 379-419.
- D. R. Stinson, On some methods for unconditionally secure key distribution and broadcast encryption. *Designs, Codes and Cryptography*, 12(1997), 215-243. 404, 408

- 23. D. R. Stinson, T. van Trung and R. Wei, Secure frameproof codes, key distribution patterns, group testing algorithms and related structures, *J. Statist. Plan. Infer.*, to appear. 401, 404
- 24. M. N. Wegman and J. L. Carter, New hash functions and their use in authentication and set equality, J. of Computer and System Science 22(1981), 265-279. 410