Lecture Notes in Computer Science 1550
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Bruce Christianson   Bruno Crispo
William S. Harbison   Michael Roe (Eds.)

Security Protocols

6th International Workshop
Cambridge, UK, April 15-17, 1998
Proceedings

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Bruce Christianson
Computer Science Department, University of Hertfordshire
Hatfield AL10 9AB, UK
E-mail: b.christianson@herts.ac.uk

Bruno Crispo
University of Cambridge, Computer Laboratory
New Museums Site, Pembroke Street, Cambridge CB2 3QG, UK
and University of Turin, Department of Computer Science
Corso Svizzera 185, I-10149 Turin, Italy
E-mail: bruno.crispo@cl.cam.ac.uk

William S. Harbison
Nortel Networks, Harlow Laboratories, London Road
Harlow, Essex CM17 9NA, UK
E-mail: wharbiso@nortelnetworks.com

Michael Roe
University of Cambridge, Centre for Communications Systems Research
10 Downing Street, Cambridge CB2 3DS, UK
E-mail: michael.roe@ccsr.cam.ac.uk

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme
Security protocols : 6th international workshop, Cambridge, UK,
April 15- 17, 1998 ; proceedings / Bruce Christianson . . . (ed.). -
Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ;
Milan ; Paris ; Singapore ; Tokyo : Springer, 1999
(Lecture notes in computer science ; Vol. 1550)
ISBN 3-540-65663-4

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

ISSN 0302-9743
ISBN 3-540-65663-4 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author
SPIN 10693156 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Preface

You hold in your hands the proceedings of the sixth Cambridge International Workshop on Security Protocols. This annual event is an opportunity for researchers in acadaemia and industry to discuss new developments in the area of distributed system security, using various insights into the general notion of a protocol as a point of departure. Although you will also find plenty of interesting new results in this volume, our primary concern is to create a forum in which researchers are free to discuss the limitations of current work: things we can't yet do, or don't yet fully understand.

This year the theme of the workshop was the interactions between trust and delegation, exploring the implications and effects of these upon such issues as authorization, security policy and system and component design. As you will see, this turned out to be a fruitful avenue. This volume marks a return to our original intention for the format of the proceedings: we bring you a short position paper from each contributor, deliberately written to provoke reasoned controversy, together with not-quite-verbatim transcripts of the discussions which ensued.

Many thanks to Stewart Lee and the University of Cambridge Centre for Communications Systems Research, for acting as hosts of the workshop, and to Roger Needham and Microsoft Research Limited (Cambridge), for providing us with an excellent venue and large amounts of coffee. This year's workshop marks twenty years since the publication of Needham and Schroeder's "Using Encryption for Authentication in Large Networks of Computers" and ten years since the first appearance of the BAN logic, so we were delighted to have Roger in the chair for the final panel discussion, a transcript of which concludes the volume.

It is a pleasure to express our gratitude to Hitachi and Nortel for providing financial support. We are also indebted to Robin Milner of the University of Cambridge Computer Laboratory for his support and assistance. Finally, we would like to thank Dorian Addison of CCSR for undertaking a Sisyphean burden of administration, and Lori Klimaszewska of the University of Cambridge Computing Service for an excellent job of transcribing audio tapes full of "techie-talk and coughing".

We use this workshop to guide our own research agendas, and we hope that engaging with the discussions in these proceedings will also provide you with some useful reflections upon your research, and maybe help inspire it in some unexpected new direction. If this happens please write and tell us, we'd love to hear from you.

Bruce Christianson
Bruno Crispo
William Harbison
Michael Roe

Contents

Inductive Analysis of the Internet Protocol TLS
Lawrence C. Paulson ... 1
Discussion ... 13

External Consistency and the Verification of Security Protocols
Simon N. Foley ... 24
Discussion ... 28

The Trust Shell Game
Carl Ellison ... 36
Discussion ... 41

Overview of the AT&T Labs Trust-Management Project
Joan Feigenbaum ... 45
Discussion ... 51

KeyNote: Trust Management for Public-Key Infrastructures
*Matt Blaze, Joan Feigenbaum, Angelos D. Keromytis** ... 59

Discussion - Trust Management ... 64

Application-Oriented Security Policies and Their Composition
Virgil D. Gligor and Serban I. Gavrila* ... 67
Discussion ... 75

Secure Fingerprinting Using Public-Key Cryptography
Hiroshi Yoshiura, Ryoichi Sasaki and Kazuo Takaragi* ... 83
Discussion ... 90

Third Party Certification of HTTP Service Access Statistics
Francesco Bergadano and Pancrazio De Mauro* ... 95
Discussion ... 100

Delegating Trust
William S. Harbison (Discussion) ... 108

Delegation of Responsibility
Bruno Crispo ... 118
Discussion ... 125

Abuse of Process
Mark Lomas (Discussion) ... 131

A New Concept in Protocols: Verifiable Computational Delegation
Peter Landrock ... 137
Discussion ... 146

Delegation and Not-So Smart Cards
Bruce Christianson and James A. Malcolm* ... 154
Discussion ... 158

Certification and Delegation
Michael Roe (Discussion) ... 168

Discussion - Differences Between Academic and Commercial Security
Virgil Gligor, Peter Landrock, Mark Lomas, Raphael Yahalom and John Warne ... 177

Optimistic Trust with Realistic *e*Nvestigators
Raphael Yahalom ... 193
Discussion ... 203

Insider Fraud
Dieter Gollmann ... 213
Discussion ... 220

Panel Session - Future Directions
Stewart Lee, Joan Feigenbaum, Virgil Gligor and Bruce Christianson
Chair: Roger Needham ... 227

Contributor Index ... 241

Inductive Analysis of the Internet Protocol TLS[⋆]
(Position Paper)

Lawrence C. Paulson

Computer Laboratory, University of Cambridge, Cambridge CB2 3QG, England
`lcp@cl.cam.ac.uk`

1 Introduction

Internet commerce requires secure communications. To order goods, a customer typically sends credit card details. To order life insurance, the customer might have to supply confidential personal data. Internet users would like to know that such information is safe from eavesdropping or tampering.

Many Web browsers protect transmissions using the protocol SSL (Secure Sockets Layer). The client and server machines exchange nonces and compute session keys from them. Version 3.0 of SSL has been designed to correct a flaw of previous versions, where an attacker could induce the parties into choosing an unnecessarily weak cryptosystem. The latest version of the protocol is called TLS (Transport Layer Security) [1]; it closely resembles SSL 3.0.

Is TLS really secure? My proofs suggest that it is, but one should draw no conclusions without reading the rest of this report, which describes how the protocol was modelled and what properties were proved. I have analyzed a much simplified form of TLS; I assume hashing and encryption to be secure.

My abstract version of TLS is simpler than the concrete protocol, but it is still more complex than the protocols typically verified. We have not reached the limit of what can be analyzed formally. The proofs were conducted using Isabelle/HOL [4], an interactive theorem prover for higher-order logic. They follow the inductive method [6], which has a clear semantics and treats infinite-state systems. Model-checking is not used, so there are no restrictions on the agent population, numbers of concurrent runs, etc.

The paper gives an overview of TLS (§2) and of the inductive method for verifying protocols (§3). It continues by presenting the Isabelle formalization of TLS (§4) and outlining some of the properties proved (§5). Finally, the paper discusses related work (§6) and concludes (§7).

2 Overview of TLS

A TLS *handshake* involves a *client*, such as a World Wide Web browser, and a Web *server*. Below, I refer to the client as A ('Alice') and the server as B

[⋆] M. Abadi encouraged me to understand TLS and referred me to related work. J. Margetson pointed out simplifications to the model. C. Ballarin commented on a draft. Research funded by the EPSRC, grants GR/K77051 'Authentication Logics' and GR/K57381 'Mechanizing Temporal Reasoning.'

('Bob'), as is customary for authentication protocols, especially since C and S often have dedicated meanings in the literature.

At the start of a handshake, A contacts B, supplying a session identifier and nonce. In response, B sends another nonce and his public-key certificate (my model omits other possibilities). Then A generates a *pre-master-secret*, a 48-byte random string, and sends it to B encrypted with his public key. A optionally sends a signed message to authenticate herself. Now, both parties calculate the *master-secret* M from the nonces and the pre-master-secret, using a secure pseudo-random-number function (PRF). They calculate session keys from the nonces and M. Each session involves a pair of symmetric keys; A encrypts using one and B encrypts using the other. Before sending application data, both parties exchange **finished** messages to confirm all details of the handshake and to check that cleartext parts of messages have not been altered.

A full handshake is not always necessary. At some later time, A can resume a session by quoting an old session identifier along with a fresh nonce. If B is willing to resume the designated session, then he replies with a fresh nonce and both parties compute fresh session keys from these nonces and the stored master-secret, M. Both sides confirm this shorter run using **finished** messages.

My version of TLS leaves out details of record formats, cryptographic algorithms, etc.: they are irrelevant in an abstract analysis. Alert and failure messages are omitted: bad sessions are simply abandoned. I also omit the **server key exchange** message, which allows anonymous sessions.

Here are the handshake messages in detail, as I model them, along with comments about their relation to full TLS. Section numbers, such as tls§7.3, refer to the TLS specification [1].

$$\textbf{client hello} \quad A \to B : A, Na, Sid, Pa$$

The items in this message include the nonce Na, called **client random**, and the session identifier Sid. Item Pa is A's set of preferences for encryption and compression. For our purposes, all that matters is that both parties can detect if this value has been altered during transmission (tls§7.4.1.2).

$$\textbf{server hello} \quad B \to A : Nb, Sid, Pb$$

B, in his turn, replies with nonce Nb (**server random**) and his preferences Pb.

$$\textbf{server certificate} \quad B \to A : \mathsf{certificate}(B, Kb)$$

The server's public key, Kb, is delivered in a certificate signed by a trusted third party. (The TLS proposal (tls§7.4.2) specifies a chain of X.509v3 certificates, but I assume a single certification authority and omit lifetimes and similar details.) Making the certificate mandatory and eliminating the **server key exchange** message (tls§7.4.5) simplifies **server hello**. I leave **certificate request** (tls§7.4.4) implicit: A herself decides whether or not to send the optional messages **client certificate** and **certificate verify**.

client certificate*	$A \to B : \mathsf{certificate}(A, Ka)$
client key exchange	$A \to B : \{\!|PMS|\!\}_{Kb}$
certificate verify*	$A \to B : \{\!|\mathsf{Hash}\{\!|Nb, B, PMS|\!\}|\!\}_{Ka^{-1}}$

I do not model the possibility of arriving at the pre-master-secret via a Diffie-Hellman exchange (tls§7.4.7). Optional messages are starred (*) above; in **certificate verify**, A authenticates herself to B by signing the hash of some items relevant to the current session. The specification is that all handshake messages should be hashed, but my proofs suggest that only Nb, B and PMS are needed.

client finished	$A \to B : \{\!|Finished|\!\}_{\mathsf{clientK}(Na,Nb,M)}$
server finished	$A \to B : \{\!|Finished|\!\}_{\mathsf{serverK}(Na,Nb,M)}$

Both parties compute the master-secret M from PMS, Na and Nb and compute *Finished* as the hash of M, *Sid*, Na, Pa, A, Nb, Pb, B. (The specification (tls§7.4.9) states that M should be hashed with all previous handshake messages.) The symmetric key $\mathsf{clientK}(Na, Nb, M)$ is intended for client encryption, while $\mathsf{serverK}(Na, Nb, M)$ is for server encryption; each party decrypts using the other's key (tls§6.3). (The corresponding MAC secrets are implicit because my model assumes strong encryption.) Once a party has received the other's **finished** message and compared it with her own, she is assured that both sides agree on all critical parameters, including M, Pa and Pb. Now she may begin sending confidential data. The SSL specification [2] erroneously states that she can send data immediately after sending her own **finished** message, before confirming these parameters.

For session resumption, the **hello** messages are the same. After checking that the session identifier is recent enough, the parties exchange **finished** messages and start sending application data. On paper, session resumption does not involve new messages. But in the model, four further events are involved. Each party stores the session parameters after a successful handshake and looks them up when resuming a session.

3 Proving Protocols Using Isabelle

Isabelle [4] is an interactive theorem prover supporting several formalisms, one of which is higher-order logic (HOL). Protocols can be modelled in Isabelle/HOL as inductive definitions. Isabelle's simplifier and classical reasoner automate large parts of the proofs. A security protocol is modelled as the set of traces that could arise when a population of agents run it. Among the agents is a spy who controls some subset of them as well as the network itself. The population is infinite, and the number of interleaved sessions is unlimited. This section summarizes the approach, described in detail elsewhere [6].

Message are composed of agent names, nonces, keys, etc.:

$\mathsf{Agent}\,A$	identity of an agent
$\mathsf{Number}\,N$	guessable number
$\mathsf{Nonce}\,N$	non-guessable number
$\mathsf{Key}\,K$	cryptographic key
$\mathsf{Hash}\,X$	hash of message X
$\mathsf{Crypt}\,KX$	encryption of X with key K
$\{\!\|X_1,\ldots,X_n\|\!\}$	concatenation of messages

Three operators are used to express security properties. Each maps a set H of messages to another such set.

– $\mathsf{parts}\,H$ is the set of message components potentially recoverable from H.
– $\mathsf{analz}\,H$ is the set of message components recoverable from H by means of decryption using keys available (recursively) in $\mathsf{analz}\,H$.
– $\mathsf{synth}\,H$ is the set of messages that could be expressed, starting from H and guessable items, using hashing, encryption and concatenation.

4 Formalizing the Protocol in Isabelle

TLS uses both public- and shared-key encryption. Each agent A has a private key $\mathsf{priK}\,A$ and public key $\mathsf{pubK}\,A$. The operators clientK and serverK create symmetric keys from a triple of nonces. Modelling the underlying pseudo-random-number generator causes some complications compared with the treatment of simple public-key protocols such as Needham-Schroeder [6].

The common properties of clientK and serverK are captured in the constant sessionK, which is assumed to be an injective source of session keys. Defining

$$\mathsf{clientK}\,X = \mathsf{sessionK}(X,0)$$
$$\mathsf{serverK}\,X = \mathsf{sessionK}(X,1).$$

ensures that the ranges of clientK and serverK are disjoint (there can be no collisions between the two).

Next come declarations of the constant tls to be a set of traces (lists of events) and of the pseudo-random function PRF. The latter has an elaborate definition in terms of the hash functions MD5 and SHA-1 (see tls§5). At the abstract level, we simply assume PRF to be injective.

Figure 1 presents the first three rules, two of which are common to all protocols. Rule *Nil* allows the empty trace. Rule *Fake* says that the spy may invent messages using past traffic and send them to any other agent. A third rule, *SpyKeys*, augments *Fake* by letting the spy use the TLS-specific functions sessionK and PRF. In conjunction with the spy's other powers, it allows him to apply sessionK and PRF to any three nonces previously available to him. It does not let him invert these functions, which we assume to be one-way. We could

Nil
```
[] ∈ tls
```

Fake
```
[| evs ∈ tls;  B ≠ Spy;  X ∈ synth (analz (spies evs)) |]
⟹ Says Spy B X  # evs ∈ tls
```

SpyKeys
```
[| evsSK ∈ tls;
   Says Spy B {|Nonce NA, Nonce NB, Nonce M|} ∈ set evsSK |]
⟹ Notes Spy {| Nonce (PRF(M,NA,NB)),
                Key (sessionK((NA,NB,M),b)) |} # evsSK ∈ tls
```

Fig. 1. Specifying TLS: Basic Rules

replace *SpyKeys* by defining a TLS version of the function synth, but we should then have to rework the underlying theory of messages.

Figure 2 presents three rules for the **hello** messages. **Client hello** lets any agent A send the nonce Na, session identifier Sid and preferences Pa to any other agent, B. The assumptions $Na \notin \mathsf{used}\, evsCH$ and $Na \notin \mathsf{range\,PRF}$ state that Na is fresh and distinct from all possible master-secrets. The latter assumption precludes the possibility that A might choose a nonce identical to some master-secret. (The standard function used does not cope with master-secrets because they never appear in traffic.) Both assumptions are reasonable because a 28-byte random string is highly unlikely to clash with any existing nonce or future master-secret.**Server hello** is modelled similarly. Its precondition is that B has received a suitable instance of **Client hello**.

The *Certificate* rule handles both **server certificate** and **client certificate**. Any agent may send his public-key certificate to any other agent. In the model, a certificate is simply an (agent, key) pair signed by the authentication server. Freshness of certificates and other details are not modelled.

The next two rules are **client key exchange** and **certificate verify** (Fig. 3). Rule *ClientKeyExch* chooses a *PMS* that is fresh and differs from all master-secrets, like the nonces in the **hello** messages. It requires **server certificate** to have been received. No agent is allowed to know the true sender of a message, so *ClientKeyExch* might deliver the *PMS* to the wrong agent. Similarly, *CertVerify* might use the *Nb* value from the wrong instance of **server hello**. Security is not compromised because the run will fail in the **finished** messages. *ClientKeyExch* not only sends the encrypted *PMS* to B but also stores it internally using the event $\mathsf{Notes}\, A\, \{\!|B, PMS|\!\}$. Other rules model A's referring to this note. For instance, *CertVerify* states that if A chose *PMS* for B and has received a **server hello** message, then she may send **certificate verify**.

Next come the **finished** messages (Fig. 4). *ClientFinished* states that if A has sent **client hello** and has received a plausible instance of **server hello** and

ClientHello
```
[| evsCH ∈ tls;  A ≠ B;  Nonce NA ∉ used evsCH;  NA
∉ range PRF |]
⟹ Says A B {|Agent A, Nonce NA, Number SID, Number PA|}
           # evsCH ∈ tls
```

ServerHello
```
[| evsSH ∈ tls;  A ≠ B;  Nonce NB ∉ used evsSH;  NB
∉ range PRF;
   Says A' B {|Agent A, Nonce NA, Number SID, Number PA|}
    ∈ set evsSH |]
⟹ Says B A {|Nonce NB, Number SID, Number PB|} # evsSH ∈ tls
```

Certificate
```
[| evsC ∈ tls;  A ≠ B |]
⟹ Says B A (certificate B (pubK B)) # evsC ∈ tls
```

Fig. 2. Specifying TLS: **Hello** Messages

ClientKeyExch
```
[| evsCX ∈ tls;  A ≠ B;  Nonce PMS ∉ used evsCX;  PMS
∉ range PRF;
   Says B' A (certificate B KB) ∈ set evsCX |]
⟹ Says A B (Crypt KB (Nonce PMS))
      # Notes A {|Agent B, Nonce PMS|}
      # evsCX ∈ tls
```

CertVerify
```
[| evsCV ∈ tls;  A ≠ B;
   Says B' A {|Nonce NB, Number SID, Number PB|} ∈ set evsCV;
   Notes A {|Agent B, Nonce PMS|} ∈ set evsCV |]
⟹ Says A B (Crypt (priK A) (Hash{|Nonce NB, Agent B, Nonce PMS|}))
             # evsCV ∈ tls
```

Fig. 3. Client key exchange and **certificate verify**

ClientFinished

```
[| evsCF ∈ tls;
   Says A  B {|Agent A, Nonce NA, Number SID, Number PA|} ∈set
evsCF;
   Says B' A {|Nonce NB, Number SID, Number PB|} ∈ set evsCF;
   Notes A {|Agent B, Nonce PMS|} ∈ set evsCF;
   M = PRF(PMS,NA,NB) |]
⟹ Says A B (Crypt (clientK(NA,NB,M))
              (Hash{|Number SID, Nonce M,
                     Nonce NA, Number PA, Agent A,
                     Nonce NB, Number PB, Agent B|}))
    # evsCF ∈ tls
```

Fig. 4. Finished messages

has chosen a *PMS* for *B*, then she can calculate the master-secret and send a **finished** message using her **client write key**. *ServerFinished* is analogous and may occur if *B* has received a **client hello**, sent a **server hello**, and received a **client key exchange** message.

That covers all the protocol messages, but the specification is not complete. Four further rules (omitted here) model agents' confirmation of a session and a subsequent session resumption.

The final rule, *Oops*, models security breaches. Any session key, if used, may end up in the hands of the spy. Session resumption turns out to be safe even if the spy has obtained session keys from earlier sessions.

Oops

```
[| evso ∈ tls;  A ≠ Spy;
   Says A B (Crypt (sessionK((NA,NB,M),b)) X) ∈ set evso |]
⟹ Says A Spy (Key (sessionK((NA,NB,M),b))) # evso ∈ tls
```

5 Properties Proved of TLS

One difficulty in protocol verification is knowing what to prove. Protocol goals are usually stated informally. Assuming an active attacker as the threat model, my proofs have addressed the TLS memo's 'three basic properties' (tls§1):

1. 'the peer's identity can be authenticated'
2. 'the negotiated secret is unavailable to eavesdroppers, and for any authenticated connection the secret cannot be obtained, even by an attacker who can place himself in the middle of the connection'
3. 'no attacker can modify the negotiation communication without being detected by the parties'

5.1 Basic Lemmas

In the inductive method, results are of three sorts: possibility properties, regularity lemmas and secrecy theorems. Possibility properties merely exercise all the rules to check that the model protocol can run. For a simple protocol, one possibility property suffices to show that message formats are compatible. For TLS, I proved four properties to check various paths through the main protocol, the **client verify** message, and session resumption.

Regularity lemmas assert properties that hold of all traffic. Nobody sends messages to himself and no private keys become compromised in any protocol step. From our specification of TLS, it is easy to prove that all certificates are valid. If certificate(B, K) appears in traffic then K is B's public key:

```
[| certificate B KB ∈ parts(spies evs);  evs ∈ tls |] ⟹
pubK B = KB
```

This property is overly strong, but adding false certificates seems pointless, since B might be under the spy's control anyway.

Many regularity lemmas are technical. Here are two typical ones. If a master-secret has appeared in traffic, then so has the underlying pre-master-secret. (Only the spy might send such a message.)

```
[| Nonce (PRF (PMS,NA,NB)) ∈ parts (spies evs);  evs ∈ tls |]
⟹ Nonce PMS ∈ parts (spies evs)
```

If a pre-master-secret is fresh, then no session key derived from it can either have been transmitted or used to encrypt.

```
[| Nonce PMS ∉ parts (spies evs);
   K = sessionK((Na, Nb, PRF(PMS,NA,NB)), b);
   evs ∈ tls |]
⟹ Key K ∉ parts (spies evs) & (∀ Y. Crypt K Y
∉ parts (spies evs))
```

Client authentication, one of the protocol's goals, is easily proved. If **certificate verify** has been sent, apparently by A, then it really has been sent by A provided A is uncompromised (not controlled by the spy). Moreover, A has chosen the pre-master-secret that is hashed in **certificate verify**.

```
[| X ∈ parts (spies evs);  X = Crypt KA⁻¹ (Hash{|nb, Agent B,
pms|});
   certificate A KA ∈ parts (spies evs);
   evs ∈ tls;  A ∉ bad |]
⟹ Says A B X ∈ set evs
```

5.2 Secrecy Goals

Other protocol goals relate to secrecy. In the inductive method, secrecy theorems concern items that are available to some agents but not to others. Proving secrecy theorems seems always to require, as a lemma, some form of session key compromise theorem. Such theorems impose limits on the message components that can become compromised by the loss of a session key. For most protocols, we require that these components contain no session keys, but for TLS, what matters is that they contain no nonces. (Nonces are of critical importance because one of them is the pre-master-secret.) The theorem seems obvious; no honest agent encrypts nonces using session keys, and the spy can only send nonces that have already been compromised. However, its proof takes over 20 seconds to run. Such proofs typically involve large, though automatic, case analyses.

```
evs ∈ tls ⟹
Nonce N ∈ analz (insert (Key (sessionK z)) (spies evs)) =
(Nonce N ∈ analz (spies evs))
```

Other secrecy proofs follow easily from the session key compromise theorem, using induction and simplification. If the master-secret is secure from the spy then so are all session keys derived from it, except those lost by the Oops rule. In fact, session keys do not form part of any traffic.

```
[| ∀ A. Says A Spy (Key (sessionK((NA,NB,M),b))) ∉ set
evs;
   Nonce M ∉ analz (spies evs);  evs ∈ tls |]
⟹ Key (sessionK((NA,NB,M),b)) ∉ parts (spies evs)
```

If A sends the **client key exchange** message to B, and both agents are uncompromised, then the pre-master-secret and master-secret will stay secret.

```
[| Notes A {|Agent B, Nonce PMS|} ∈ set evs;
   evs ∈ tls;  A ∉ bad;  B ∉ bad |]
⟹ Nonce PMS ∉ analz (spies evs)

[| Notes A {|Agent B, Nonce PMS|} ∈ set evs;
   evs ∈ tls;  A ∉ bad;  B ∉ bad |]
⟹ Nonce (PRF(PMS,NA,NB)) ∉ analz (spies evs)
```

5.3 Finished Messages

The most important protocol goals concern authenticity of the **finished** message. If each party can know that the **finished** message just received indeed came from the expected agent, then they can compare the message components to confirm that no tampering has occurred. Naturally, these guarantees are conditional on both agents' being uncompromised.

The client's guarantee states that if A has chosen a pre-master-secret PMS for B and if any **finished** message is present that has been encrypted with a **server write key** derived from PMS, and if B has not given that session key to the spy (via *Oops*), then B himself has sent that message, and to A.

The server's guarantee is slightly different. If any message has been encrypted with a **client write key** derived from a given PMS—which we assume to have come from A—and if A has not given that session key to the spy, then A herself sent that message, and to B.

```
[| M = PRF(PMS,NA,NB);
   Crypt (clientK(Na,Nb,M)) Y ∈ parts (spies evs);
   Notes A {|Agent B, Nonce PMS|} ∈ set evs;
   Says A Spy (Key(clientK(Na,Nb,M))) ∉ set evs;
   evs ∈ tls;  A ∉ bad;  B ∉ bad |]
⟹ Says A B (Crypt (clientK(Na,Nb,M)) Y) ∈ set evs
```

The assumption (involving Notes) that A chose the PMS is essential. If A has not authenticated herself, then B must simply trust that she is present. By sending **certificate verify**, A can discharge this assumption:

```
[| Crypt KA⁻¹ (Hash{|nb, Agent B, Nonce PMS|}) ∈ parts (spies
evs);
   certificate A KA ∈ parts (spies evs);
   evs ∈ tls;  A ∉ bad |]
⟹ Notes A {|Agent B, Nonce PMS|} ∈ set evs
```

B's guarantee does not even require his inspecting the **finished** message. The very use of `clientK(Na,Nb,M)` is proof that the communication is from A to B. If we consider the analogous property for A, we find that using `serverK(Na,Nb,M)` only guarantees that the sender is B; in the absence of **certificate verify**, B has no evidence that the PMS came from A. If he sends **server finished** to somebody else then the session will fail, so there is no security breach. Still, changing **client key exchange** to include A's identity,

$$A \to B : \{\!|A, PMS|\!\}_{Kb},$$

would slightly strengthen the protocol and simplify the analysis.

The guarantees for **finished** messages apply to session resumption as well as to full handshakes. The inductive proofs cover all the rules that make up the definition of the constant tls, including those that model resumption.

6 Related Work

Wagner and Schneier [7] analyze SSL 3.0 in detail. Their form of scrutiny, particularly concerning attacks against the underlying cryptosystems, will remain an essential complement to proving protocols at the abstract level.

In his PhD thesis, Sven Dietrich analyses SSL 3.0 using the belief logic NCP (non-monotonic cryptographic protocols). Recall that SSL allows both authenticated and unauthenticated sessions; Dietrich considers the latter and shows them to be secure against a passive eavesdropper. Although NCP is a formal logic, Dietrich appears to have generated his lengthy derivations by hand.

Mitchell et al. [3] apply model checking to a number of simple protocols derived from SSL 3.0. Most of the protocols are badly flawed (no nonces, for example) and the model checker finds many attacks.

7 Conclusions

The inductive method has many advantages. Its semantic framework, based on the actions agents can perform, has few of the peculiarities of belief logics. Proofs impose no limits on the number of simultaneous or resumed sessions. Isabelle's automatic tools allow the proofs to be generated with a moderate effort, and they run fast. The full TLS proof script runs in a few minutes.

I obtained the abstract message exchange given in §2 by reverse engineering the TLS specification. This process took about two weeks, one-third of the time spent on this verification. SSL must have originated in such a message exchange, but I could not find one in the literature. If security protocols are to be trusted, their design process must be transparent. The underlying abstract protocol should be exposed to public scrutiny. The concrete protocol should be presented as a faithful realization of the abstract one. Designers should distinguish between attacks against the abstract message exchange and those against the concrete protocol.

All the expected security goals were proved: no attacks turned up. This unexciting outcome might be expected in a protocol already so thoroughly examined. No unusual lines of reasoning were required, unlike the case of the Yahalom protocol [5]. The proofs did yield some insights into TLS, such as the possibility of strengthening **client key exchange** by including A's identity (§5). The main interest of this work lies in the modelling of TLS, especially its use of pseudo-random number generators.

In several places, the protocol requires computing the hash of 'all previous handshake messages.' There is obviously much redundancy, and the requirement is ambiguous too; the specification is sprinkled with remarks that certain routine messages or components should not be hashed. One such message, **change cipher spec**, was thereby omitted and later turned out to be essential [7]. I suggest, therefore, that hashes should be computed not over everything but over selected items that the protocol designer requires to be confirmed. An inductive analysis can help in selecting the critical message components. The TLS security analysis (tls§F.1.1.2) states that the critical components of the hash in **certificate verify** are the server's name and nonce, but my proofs suggest that the pre-master-secret is also necessary.

Once session keys have been established, the parties have a secure channel upon which they must run a reliable communication protocol. Abadi tells me that the part of TLS concerned with communication should also be verified, since this aspect of SSL once contained errors. I have considered only the security aspects of TLS. The communication protocol could be verified separately, assuming an unreliable medium rather than an enemy. My proofs assume that application data does not contain secrets associated with TLS sessions, such as keys and master-secrets; if it does, then one security breach could lead to many others.

References

1. Tim Dierks and Christopher Allen. The TLS protocol: Version 1.0, November 1997. Internet-draft `draft-ietf-tls-protocol-05.txt`.
2. Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL protocol version 3.0.
3. John C. Mitchell, Vitaly Shmatikov, and Ulrich Stern. Finite-state analysis of SSL 3.0 and related protocols. In Hilarie Orman and Catherine Meadows, editors, *Workshop on Design and Formal Verification of Security Protocols*. DIMACS, September 1997.
4. Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*. Springer, 1994. LNCS 828.
5. Lawrence C. Paulson. On two formal analyses of the Yahalom protocol. Technical Report 432, Computer Laboratory, University of Cambridge, July 1997.
6. Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 1998. in press.
7. David Wagner and Bruce Schneier. Analysis of the SSL 3.0 protocol. On the Internet at `http://www.cs.berkeley.edu/~daw/ssl3.0.ps`, 1996.

Inductive Analysis of the Internet Protocol TLS (Transcript of Discussion)

Larry Paulson

Computer Laboratory, University of Cambridge

Although I work by proving the correctness of protocols using formal proofs, I would never say that you get there by iron-clad guarantee because one is always simplifying the world, one is always assuming that if you have say two forty-byte nonces being chosen then they will definitely never ever coincide which of course is an oversimplification, and when you make a lot of such assumptions there is a margin of error there so I'm not even sure it's possible in principle to get perfection.

Anyway I've been looking at the Internet protocol TLS and I don't know how well known that is. Has anyone heard of SSL which is used in things like Netscape? Well TLS is the next version of SSL, Martín Abadi told me about it, and in fact at my level of analysis it is SSL but it has a rather nicer looking definition than SSL did. I think the main lesson of this, of my talk, is that here we have a deployed protocol of some complexity, you know it does a lot more than Needham and Schroeder and I wouldn't claim that I have analysed everything in SSL, it's a jolly complicated protocol, it had all these layers, this record layer which I didn't even look at which Martín Abadi told me had been very broken at one point, but there's still a lot left that I did examine.

So just briefly what it does, in case you don't know, you use it in Netscape if you were sending a secure form, to protect the form from eavesdropping and tampering, it uses both public and symmetric key encryption and it has a couple of unusual features, at least if you've been looking at protocol verification. Almost always they work by somebody thinks up a session key and distributes it, whereas this works more like a Diffie Hellman sort of thing. The two parties calculate, they exchange nonces, and then they calculate the session key and rather less work has been done on looking at protocols to do with that kind of thing. Another awkward feature of this protocol is you're allowed to short circuit parts of it so if you've already had a connection a while ago and you want to resume the connection you can short circuit the protocol and go straight into a new connection and naturally one wants to know if that's secure or not. The main thing to look at is there are something like I believe eight bits and pieces that can go to and fro. Now all the dashed lines like there, there and there, are optional. There are many paths through this protocol and there's a lot of toing and froing.

I suppose I should give you a brief snapshot of what goes on. The two parties which I typically do call Alice and Bob because we all do even though they're a client and a server, they introduce each other, they exchange nonces, they have a session identifier which is not meant to be cryptographic in any way which they also will send, then they exchange some public key certificates, and then

Alice will send a long secret nonce which is used to generate session keys, and finally they send these two messages at the end to confirm everything. Basically it's the usual explicitness idea of hashing up everything you've got and sending it back so that both sides can confirm it's OK. And roughly, how it works, if you use the full protocol then both sides are meant to authenticate themselves to the other. In Alice's case, she authenticates herself by signing something. Bob only authenticates himself in the indirect way in that he appears to be able to read things Alice has sent to him under his public key and therefore it must be Bob.

Now this protocol has gone through many versions, basically because all the previous versions were attacked. As far as I know, those last two messages were put in because the initial messages are sent in clear of course, and it turned out, that among the options in there is you can say what kind of crypto system you want to use, and you could send a message saying I want to use extra strong iron clad crypto system and your malicious person could turn that to I want to turn this into doddle easy to break crypto system and that tampering was not detected until later versions.

Here is a little more detail. The first two messages where the two people wake up each other. This is the Netscape browser and that's the web server saying I want to send you something. And one interesting thing here, this is where session resumption can occur. You saw there this long exchange, however, I never quite understood this so don't ask me, I believe that it's the case that if Alice reuses a session ID and if Bob says oh you gave me that session ID twenty minutes ago then it will be assumed that this is a request for a resumption of the session in which case you skip the whole rest of the protocol and go to the end. Now you will say what happens if the old session keys have been compromised and it turns out, as far as I can tell, that it is still secure because you compute new session keys, even with this short circuited version, you still get fresh session keys and it still appears to be secure. If you don't do the short circuit then you start exchanging certificates and so on. Now I said that one has to oversimplify and I got this giant Internet Draft which I gather is the usual way one finds these things, and I can tell you it wasn't easy to decipher. I gather that a certificate was a very complicated thing and I couldn't work out what it was so I said right we assume we know what a certificate is, and about chains of certificates, and expiry dates, and trusting the person who signed it. I assumed this to be an atomic act. So one of the biggest holes perhaps in this analysis is that if there is some tricky thing you can do to with bogus certificates then my analysis is not going to pick that up. You can actually prove a theorem in my system which says that all certificates are perfect.

So now a few more. I think I'd like to have a dig at model checking people because it's a competitive world out there of course. I actually think model checking has been very successful, however, one has to stand up for one's own side. Someone told me that it's harder for model checking if you have lots of options and this protocol certainly has lots of options so this method is optional, that method is optional, and of course with resumption other big parts are optional. And the thing one is tempted to do and indeed a thing that I did at first, was first to say they're not optional and second to say I don't like having lots of messages because look there are three messages from Alice to Bob, now why can't you just send one big message. And people who look at this using model checking did that, they made everything mandatory and jammed all the messages into one giant message. I discovered later, because I did the same thing, that this was quite dumb at least in my approach, because for me the cost of an analysis is roughly linear in the number of messages but it can be exponential in the complexity of a single message, so a giant message is the worst thing you can do. And if you're looking for ways to break my approach just think of a protocol which you have nested encryption going three or four deep; I don't if I could cope with it because there is a doubling of the effort with every encryption.

I don't think it would be helpful to go through all the details of this but maybe just one thing that is very important. I assume you all know how Diffie Hellman type things work. This isn't Diffie Hellman but it's the same idea. This is a nonce of something like 40 or 48 bytes long, it's meant to be random, cryptographically secure, etc. and both parties will hash this up with the two other nonces, one of which they picked themselves, so each has a say in the final session key, and clearly this has got to be kept secret because if it gets out then they've let the cat out of the bag completely. And it's called PMS which stands for pre-master secret.

One of the biggest things I didn't look at was that you can do a Diffie Hellman exchange, a real Diffie Hellman exchange, if Bob doesn't have a public key I think, they can do a Diffie Hellman exchange instead to get this thing across. Now for my sort of person the confusing thing is that you will read the protocol description and they will say you can do all these things and they never sort of say that if you do a Diffie Hellman exchange then you are vulnerable from an active attacker because perhaps they assume everybody knows that. They never really say what the threat model is, they say informally that while we would like things to be secure from being read and all that kind of thing, but they never say secure against what, which is sort of what Roger was hinting at that you need to have clear in your mind what sort of tampering you're trying to defeat. And I had to infer all of that from the specification.

Now to show you the end of the protocol there are one or two other interesting features. Now I've never seen this before, but maybe you have. Although they use symmetric key encryption they still have two session keys for every connection, maybe this is a well-known thing, so that Alice has her key which she uses for encryption and which Bob uses to decrypt and Bob has his key which he uses to encrypt and Alice uses to decrypt. Unless this is an obvious technique, I will have to guess why this is done, maybe that having two session keys makes it a little harder to break than having one.

Michael Roe: Maybe it could be used as protection against reflection where you take a message from Alice to Bob and turn it round and make it look to Alice like a message from Bob.

Reply: Well that is certainly needed here because the content of these messages are the same, so reflection would otherwise work.

And there was a bug even in the latest specification of SSL saying that after Alice sends her message she can immediately send all kinds of secret data and that is a bug because she must not do that until until Bob has reflected the same thing back to her and then finally when she's checked it, only then is it safe.

I should maybe say a little about these calculations, just briefly. There is the pre-master secret which is a big nonce, it's scrambled up with two other nonces to get this thing and the master secret, another big nonce, and that thing (M), the master secret, is then scrambled up with the same nonces again. Now that looks rather dumb, why do you scramble the nonces twice, and the reason is if you resume a session then the M will be an old one which you've stored and then looked up but the nonces will be fresh ones. Every time you establish a new connection you have new nonces so you'll get fresh session keys with your resumption, and that's how the resumptions are made more secure. You wouldn't want to use the old session keys again. OK I hope that is clear enough.

Now the way I look at this, and I do need a disclaimer, I'm not using a BAN-type logic and I'm not using model checking, so I should say it's the third way. I'm using higher order logic and I'm using a lot of set theoretic things, relatively elementary things like union and intersection, I do try and plug them, you could do this in an untyped set theory if you prefer, and I'm doing things in a fairly obvious concrete way so you merely have a formal model in which you put in all the agents and all the things you expect them to do. When I say expect, the things they could do, no-one is ever forced to do anything. But now of course Roger was saying that in some protocols you might assume that everybody is crooked, I am not taking that approach here although in fact I have an arbitrary set of crooked people around. There is a Satan there and Satan has a number of slaves who look like ordinary users and then you have an honest population as well, and these people all behave as you would expect the honest people behave honestly, although they do make mistakes sometimes and the behaviour of Satan is not completely chaotic but he does everything he can reasonably be expected to do without supernatural powers, which is him, he is the active attacker. He does the obvious things, eavesdrops on messages, decrypts things with keys he's got, puts other things together and sends them out to random people and so on. What shall I say here, I mentioned that even the honest ones are careless, I can describe how that's formalised later, but basically they occasionally lose information and what you want to know is that losing some bit of information like losing one session key say, you don't want it to collapse the whole system.

I mechanised this using Isabel which is a tool I've developed myself, and it's never been clear in my mind whether I did this merely to publicise Isabel or whether I thought it was something I could do with Isabel and it would be fun. Anyway, the model is fairly straightforward so you say we have messages that are built up of a lot of fairly obvious things like names of agents, unguessable nonces, i.e. long random ones, guessable ones such as sequence numbers, time stamps and so on, you can hash things, there are keys, fairly obvious things that one can include in a message. And naturally if you have all these items you can string them together to form compound messages. And one assumes pretty much the obvious properties of these. Another weakness of this system if you're looking for them is that encryption is assumed to be very strong so if something is encrypted you cannot do anything at all to it unless you've got the matching key. You can't, for example, turn it into another intelligible message even. Not only can you not read it, but you can't tamper with it.

Since I don't want to run out of time, I don't want to go through all the formal rules in the model, but I'll just give you one which will give you the flavour of how all the things are done. You see I'm modelling what agents can do and if the honest agents act normally, if they're in a certain stage in the protocol and if they think they're entitled to move to the next stage, they'll move to the next stage. And the only way they can know that they're at a certain stage is because they've certain things in a network, they don't know where they came from, but they'll believe they came from whoever they seem to have come from, and now let's carry out the next step. So, for example, one of the points in the protocol is where Alice sends out this pre master secret, and she does it if she has previously received a message which she thinks came from Bob and also if she received a public key certificate from Bob. Now she doesn't know where they really came from which is why I put the B prime and B double prime there, she can't know the true sender of course, but if she's got this she might invent a pre master secret and I give her the power to actually pick one that's never been used before, so this is one of those assumptions that she will never get an unpleasant coincidence, and then she will send it to Bob. Clearly you don't write English in an Isabel specification, but you write something that's pretty close to this. And now this is hardly non-deterministic because in fact this gives, there is nothing in the way I modelled it that forces the thing to only happen once, for example, or ever to happen, and there is no real connection between Bob and any of this, so in fact any session key that Alice has ever seen in her life, or any certificate and any public key that's enclosed in the certificate, she might use to send out pre master secrets to all and sundry. Now you might say my model is too loose but in fact it's better that way because it simplifies the formalisation, and if you can still prove the thing you want, even in this very sloppy protocol in which people are sending out master secrets all over the place, then it will, the protocol will still be secure even if you stop all those silly things from happening. There is an awful lot of this non-determinism in there.

I suppose these parts of the model are important. This is a general thing that one always has if you've got an active attacker. There is this person called spy, there is a rather elaborate way of describing things the spy can do, given past traffic and whenever he is capable of forging any message he might do it and send it to anybody. So this is always trying to put a spanner in the works. And there is an additional thing needed for this protocol because here you have these functions, this PRF stands for pseudo random function which is sort of glorified hash function, and this other one which is another glorified hash function, which take arguments and scramble them up and do something with them. You have to give the spy the power to apply these because otherwise clearly he's much too weak. If the spy gets hold of a master secret then he can invent the session keys and so on. So you have to give him that power.

And finally, there is the carelessness I told you about. Any agent in the course of using a session key might hand it over to the spy at the same time, so I have found you need to build carelessness into the model because if you don't you get a much too strong system. Just as, you know, if the bank could assume all its tellers were honest then, you know, life would be easy.

Now I've proved a lot of grand sounding things about TLS so I almost want to say it's correct although I know what a dangerous thing that is to say but let me just say I haven't found anything wrong with it and to the level of model that I've got it seems to be pretty strong. So that pre-master secret which was a nonce sent by Alice seems to stay secret, this is not actually a very surprising result. It's sent once, it's encrypted and then it's never sent again. The master secret remains secret, maybe that isn't too surprising either because both sides calculate this thing and never send it anywhere so it couldn't be compromised without physically breaking into their computer. If Alice authenticates herself, she doesn't have to, now if she doesn't authenticate herself then of course there are many attacks, middle person attacks could then occur, but assuming that Alice does sign something using her public key as the protocol lets her do then of course she's really there and the session keys remain secret so when you are talking using the session keys that have been calculated then they really are protecting the messages and we get a kind of authentication. If you receive a message encrypted with a session key that should be Alice's session key, then it really did originate with Alice. Now I told you there is some carelessness in the model and of course these guarantees are subject to the condition that that particular key hasn't been given to the spy. But the useful thing there is even if the spy holds a lot of other session keys, including session keys from earlier instances of this session, for the session identifier, the current session keys will still be OK.

To prove those results I had to prove a lot of technical lemmas, I don't want to talk about all of them in too much detail, but there were some of these, you get a lot of interaction between the different parts of protocol messages and this is where if I want to say that this better than model checking, it's better and worse. With model checking, you model it, you press a button and it's checked, and it's checked without proving any lemmas, you just check the thing you want and bingo, just like that. Now the one drawback of model checking is perhaps that model has to be simpler than you'd like but the other point I would like to make about formal proof, it does take some work. Now TLS took my about six weeks altogether, so there is some work involved, but in part of that work you actually get to understand what the protocol's doing which I don't think you get from a model check, a model check just checks it. Here you can actually see how the different messages, now this could be that but it doesn't because of something else, and you can then use that information, oh I don't know, it's useful if you're designing the next protocol at least.

I don't want to spend a lot of time on related work I guess because I want to leave a little time for questions, but there have been three other studies. There's been a rather informal cryptographic sort of one which I think will always be necessary because it's at a lower level than mine looking at attacks against the crypto systems. There's been one using a very sort of hyped up BAN logic, a sort of hyper exponential BAN logic I would call it, in which it's like sixty pages of derivation instead of six lines like the original BAN paper and there's been some model checking done by John Mitchell and some students.

I don't have a lot of time to talk about this proposed tiny protocol change but with everyone it seems to be a thing that you do: when you analyse a protocol, you suggest a change. Actually the benefit you get from this is ϵ and I was never able to figure out how big in bits it cost to put an agents name in there and no-one can tell me things like how big is an agents name so I don't know, if one of you knows the answer maybe tell me some time, but it's strengthened very slightly the analysis and the properties you get from the protocol. But this isn't quite the same as an explicitness requirement that you would have another protocol because everything is checked later when you calculate things so you this isn't needed to deter any actual attacks.

I want to skip this probably due to lack of time, but I do want to make a brief sermon. I got this TLS horrible 150 page Internet Draft and Martín Abadi fortunately came over and worked through it with me and helped me understand what was going on. And I do think that protocol designers could actually say this is my abstract message exchange, this is part of the formal definition, it's not merely a pretty diagram that you can look up, but I actually got it as part of the specification and I will then claim that my bit level specification faithfully implements the high level one which one could then verify ...

Joan Feigenbaum: Do part of the proof in the spec?

Reply: No, no, I'm not asking them to do part of the proof, I'm merely saying this is an abstract message, saying that we guarantee it is a faithful representation of the bit level.

Joan Feigenbaum: Which we *guarantee*?

Reply: Well they guarantee in that, well ...

Joan Feigenbaum: Because that's the hardest part of the proof, saying that what you implement ...

Reply: When I say guarantee, what I mean is, I clearly don't know one hundred percent guarantee, but they can at least try to make it so, and not have the figure being unchanged since the first version or something like that.

There was a pretty picture in the spec but I couldn't really get anything from it, it didn't really say enough, but I didn't know whether it was really a convincing picture or not. So I took two weeks, two weeks was just getting my own picture, that was all, and that's a third of the effort was just devoted to working out what the spec was. It takes three minutes for the proof scripts to run so the computational effort is pretty modest, and the human effort is tolerable. I think it would take longer to implement TLS than to prove it at this level, so that's not too bad. The proofs were actually not very exciting, Yahalom is a much more interest protocol to analyse, but it's interesting that one can model these other things. I think I should stop and leave at least one or two questions.

Bill Harbison: Larry thank you very much. Well you seem to have stimulated a little bit of discussion already.

Raphael Yahalom: I was wondering regarding the earlier SSL bugs which you mentioned, do you have any indication or any feeling as to how many of these you would you have found if you had attacked that protocol? Have you analysed that protocol?

Reply: I only know in detail of one earlier bug which was the non-confirmation of the protocol options and I think, I don't want to make glib remarks because I would probably of said of these protocol options, oh there just different kinds of encryption and encryption is perfect therefore I don't care what kind of encryption is chosen. I have always found that it is so very slippery, it's so easy to misinterpret either to get the model slightly wrong so that it's useless or to misinterpret one's results, so I wouldn't want to claim I would have found it, one might have found it.

Audience: How unfortunate was the assumption of the cryptography being not malleable, that you can't tamper?

Reply: I think one can change the model. It complicates things a bit and I think one clearly has to look at that kind of thing because there are a lot of things like public key encryption (or xor for that matter) where that would happen, but I think in this case, I think it is not a bad assumption. I can't recall exactly how RSA is used. I assume it's protected by some sort of explicitness so that you couldn't turn these messages into other things.

Michael Roe: There are real protocols that have failed because the protocol designer hasn't understood the difference between confidentiality and integrity and has used the transformation suitable for one where the other was needed.

Reply: Yes, clearly there are many, many directions in which one can still extend this work and those are some of them.

Bill Harbison: Larry I have a question somewhat related to Rafi's. Those of us who have had to try to construct real systems in the real word from specifications, even those which are less than 150 pages long, find that 90% of the work is in fact in just interpreting the specification. What is also our experience, and I will speak for many here, is that two different people will not arrive at the same precise understanding of what that specification is. My question to you is, how much do you think of what is in your analysis of TLS reflect your going through the specification and interpreting non-explicitly in your assumptions, how it operates?

Reply: You mean I was inferring what correctness would mean for this? I had to in that one saw this mechanism for generating session keys that I didn't understand at first which wasn't explained anywhere. It seemed at first hopelessly complicated and silly, and it was during the course of, or maybe even discussing it with somebody, that I suddenly realised that the nonces were hashed in twice because if you resumed a session then you would have different nonces and so you get different session keys. It was certainly never pointed out in the introduction that they do this in order to protect session resumption. Indeed, only the very latest thing had this, the obvious thing was for every protocol message it would say what this message is for and when this message is sent. You would think that that would be the first thing they would include but in fact it was the last thing that then got in the very final draft.

Michael Roe: What it's for is often hard because that's a complicated global property of the whole protocol. Who sends it and what they do with it is the kind of thing you ought to expect to see in there!

Reply: Oh yes, you would like to know when, I often would puzzle over whether the message was optional or not, I'd spend ages trying to work it out.

Peter Landrock: What exactly did you mean by the statement that all certificates are perfect?

Reply: In my system, what does that mean? It means that if you have a certificate saying that Bob's public key is K then Bob's public key really is K.

Michael Roe: You put that in as an axiom because you haven't wanted to model the entirely separate issues of the certificate part.

Reply: That's right, but it is true that Bob might be a crook so even though all certificates might be perfect, it isn't meant to be a perfect world.

Bill Harbison: I thought you were going to ask about repudiation there Mike.

Bruce Christianson: Can I ask if you could say just a little about why the kitchen sink shouldn't be hashed?

Reply: OK sure, in TLS they had this bug and perhaps it was an overreaction to this, they said right we're going to have these messages in which we hash absolutely everything and for explicitness sake, and send it along. And maybe it occurred to somebody that there was a little redundancy in this because many of the messages already had been repeating bits of other messages so there were all little exclusions all through the things saying the following message doesn't count as a message for the purposes of this hashing. It was very hard to work out what was meant to be hashed and in fact from my point of view it was much easier just to think up particular components and hash those explicitly rather than hashing previous messages. One of the messages which they decided didn't really count as a message, turned out to be a security critical message and then by leaving it out they actually let in an attacker. So I think one should be explicit about one's explicitness in other words if a particular thing needs to be confirmed like the encryption options then you should say right hash the encryption options. Just to say hash the whole lot is being lazy I think.

Virgil Gligor: You brought up the notion of model checking several times and your first line seems to imply that the only difference between your approach and model checking is one of efficiency of time.

Reply: Oh no, no they're different in every way.

Virgil Gligor: So my question is are there any security properties that you looked at here that are not model checkable?

Reply: I would never want to make such a claim because there are a lot of very clever people out there running model checkers, and when I want to make the case for my own method, because the model checking field is a very strong one, I say first that one needs different methods anyway, and second the greater ability to understand the interaction between parts of a protocol and there will be things that perhaps I can model and they can't and there may be things that they can model much more quickly because of course they have this great advantage of speed, not in terms of raw speed necessarily, but certainly they don't have to spend ages trying to prove things. Once they've modelled it, it's checked, you know in a day. It is very hard to say that about formal proof.

Audience: I noticed in your paper you suggest that the Wagner and Schneier approach and your approach are complimentary. Are you suggesting in that statement that you should apply both approaches in order to increase confidence in one's results.

Reply: Oh, certainly, they're working on different levels so Wagner and Schneier say that if someone attempts repeated session resumption they'll get lots of cryptotext with known plaintext which is something that she could use to attack the cryptosystem. Now my analysis is assuming encryption to be perfect so it's completely silent on that whole obviously important area of security.

Michael Roe: I'd like to raise the usual mathematicians' objection against machine generated proofs. If this had been a BAN proof we'd have seen it up there on the overhead projector transparencies then we could all have fun checking it and working out what the underlying assumptions here, but as soon as you start getting into these machine proofs you have not a proof but the assertion that a proof exists. And the assumptions underlying the proof have never been made entirely apparent.

Reply: It is not that bad, however. To understand the proofs you have to understand two things, one is the model and of course well you look at the model in detail you might take issue with it. I have tried to point out the main simplifications in the model. As for the proofs, I don't think you need to see the proofs in utter gory detail, I think machine proofs are much longer internally than a human proof because a machine is dumb and does everything it can. The think I like about the model generally, and the reason I arrived at it in the first place is I saw how people argued about it and they'd say well this secret can't get out because it couldn't get up here and it couldn't get up there and, you know, they're arguing to try the earliest place where the secret got out and to me that's inductive reasoning and that's why I've chosen an inductive model because it fits in formal reasoning. When you look at one of these proofs and say what's really going on you can say right we have case analysis on all the eight protocol messages, now they look at message five, could it have got out there and then you can actually see a case analysis well either Bob is crooked or he's honest, if he's crooked then something, if he's honest then something else. So you can explicate these two.

Michael Roe: That's true for analysing a protocol for yourself , I agree these things are very good but there seems to be something that falls out in the methodology for convincing other people that the protocol is OK. I mean when you say here's a proof of it and the proof is open for examination, that carries conviction whereas the assertion that a program run on a computer at some time really did print out a yes answer is less satisfactory.

Stewart Lee: It's a question of trust.

Reply: Well, you've introduced a question of trust that wasn't there before. To make them good for public consumption, turn the machine proofs into hand proofs, proof sketches with the attempt of convincing people. I think if you look at my paper on the Yahalom protocol there is something of a stab at that in which it presents rather intricate connections of things that finally make this protocol work. I do try and give a convincing informal argument as to why it holds together.

Bruce Christianson: I think one of the reasons why people are sometimes a bit reluctant to give the kind of abstract model that you've indicated would have been a big help, is because people are well aware there's more than one way to model and a particular party's preoccupations about the threat models that they've perceived as being relevant to them influences that choice of model quite a lot. So lots of us get very reluctant to have an authorised version of the model because we say well we want to stand or fall on whether the protocol works, we don't want to get shot down in flames because we haven't picked your favourite model. And I think Mike's sort of unpicking that reluctance from the other end.

Reply: I didn't mean a formal model, I just meant that someone should say these are the messages that can be exchanged in TLS, these are mandatory, these are optional, these are the components, without saying how many bits long they were, or cryptosystem is used. And that would have been very helpful. And assumptions about the cryptosystem. I really don't know if they need the encryption to be strong or not in TLS. I still don't know.

Bruce Christianson: But I think one of the nice things about this approach is that now for the first time it is possible to say alright here's the model I'm using, here's the formal proof, now you can take the model and play with it and get a formal proof for yourself and then you don't even need to show me what you've done to convince yourself that this protocol works, but once we're both convinced it works we'll start using it.

External Consistency and the Verification of Security Protocols
(Position Paper)

Simon N. Foley

Centre for Communications Systems Research
University of Cambridge
Cambridge CB2 3DS, UK.

Abstract. The notion of external consistency—that system state correctly reflects the real world—provides a basis for a denotational definition of integrity. We regard segregation of duties, well formed transactions, auditing, replication, MACs, and so forth, as simply implementation techniques: they define *how* to achieve this notion of integrity in an operational sense. Therefore, we argue that when a designer claims that a system is fault-tolerant, or that a protocol properly authenticates, or that a system is secure against fraud, then what the designer is actually claiming is that it is externally consistent. An advantage of taking this view is that it allows us to give a meaning to the 'security' of a system that uses a combination of these implementation techniques.

1 A Systems View

Requirements Analysis [10] typically identifies the *essential* functional requirements that define *what* the system must do. An implementation defines *how* the system operates and must take into consideration the fact that the infrastructure that is put in place to support the the requirements may be unreliable. For example, an infrastructure should include a backup and restore subsystem. While not part of the essential requirements, it is a necessary part of the implementation since the infrastructure can corrupt data.

Example 1. When a consignment arrives from a supplier (Figure 1) a clerk verifies it against its accompanying consignment note (cnote). This clerk, part of the infrastructure, uses the cnote to enter invoice details (inv) to a computer system, which in turn, sends payment (pay) directly to the supplier.

The essential requirement for this simplistic *enterprise* is that on receipt of a cnote, a pay should be generated. External Consistency (integrity) is maintained if, even in the presence of failures within the infrastructure, the enterprise can still support this requirement at it's external interface with the supplier. △

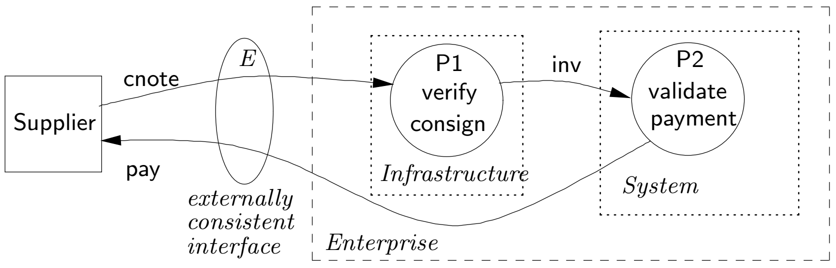

Fig. 1. Data flow diagram of a simple payment system

2 External Consistency

We assume that an enterprise is composed of a reliable and unreliable components. If external consistency is to be maintained then the reliable part of the enterprise must be capable of compensating for the unreliable section.

Suppose that the behaviour of payment enterprise is described by the composition of $\mathsf{P1}$ and $\mathsf{P2}$. Under normal operating conditions the clerk alternates between processing cnotes and invs. As a result, the $\mathsf{Supplier}$ participates in a sequence of cnote and pay messages through its interface with the enterprise. For example, the supplier could expect to engage in a sequence $\langle \mathsf{cnote}, \mathsf{pay} \rangle$ with the enterprise, but would not expect, for example, to engage $\langle \mathsf{pay}, \mathsf{pay} \rangle$.

However, it is possible that the clerk can fail and as a result take on an arbitrary behaviour $\overline{\mathsf{P1}}$, corresponding to an infrastructure failure. External consistency is achieved if the behaviour at the external interface E, resulting from an infrastructure failure, is consistent with the behaviour under normal conditions. Formally,

$$(\overline{\mathsf{P1}} \| \mathsf{P2})@E \subseteq (\mathsf{P1} \| \mathsf{P2})@E$$

where $\mathsf{P1} \| \mathsf{P2}$ is the parallel composition of (processes) $\mathsf{P1}$ and $\mathsf{P2}$ [6] and $P@E$ is the set of observations that can be made at the interface E of process P. This is an information-flow style definition (in the sense of [1]): variety in the behaviour (normal or abnormal) of the infrastructure may not be conveyed to the interface. A formal study of this definition, and the examples in this paper may be found in [5].

3 Implementing External Consistency

It's clear that the payment subsystem, as currently described, is not externally consistent at interface E: an (abnormal) clerk can engage in arbitrary inv events, generating a stream of payments at the external interface that do not have corresponding cnote events (as per a normal behaviour).

If the system and supplier share a secret key (unknown to the clerk) then a Message Authentication Code (MAC) can be included in cnote to ensure the authenticity of the message. In this case, the abnormal behaviour of the clerk is limited in that he cannot construct arbitrary MACs. This limitation must be characterised in the specification of possible abnormal behaviour $\overline{\mathsf{P1}}$. Assuming that the design of the system $\mathsf{P2}$ is such that it is resilient to forged and replayed inv messages, then we argue that external consistency is achieved at interface E. This approach would appear to be consistent with existing protocol analysis techniques, for example, [4,7,8], which use a roughly similar strategy to validate authentication protocols. The protocol is specified as $\mathsf{P2}$ and the attacker (Spy) makes up the infrastructure, choosing to have normal ($\mathsf{P1}$) or abnormal ($\overline{\mathsf{P1}}$) behaviour. The key difference is that [4,7,8] are tailored to specific theorem-proving or model-checking environments.

An alternative (but not practical) approach to implementing integrity in the payment subsystem is to make the infrastructure fault tolerant. Assume that every consignment is processed simultaneously by $2n + 1$ replicated clerks; the system votes on the ($2n+1$ invoices) to decide whether a consignment is valid. In this case, the abnormal behaviour of the infrastructure is represented by at least $n + 1$ clerks having normal behaviour, and we argue that external consistency is achieved at the interface E. This is consistent with existing results where non-interference techniques have been used to verify fault-tolerance [9,11].

A further alternative to implementing integrity is to use segregation of duties. Assume that a separate invoice and consignment note is used for every consignment, and these are processed by different clerks. In this case the abnormal behaviour of the infrastructure corresponds to one clerk behaving abnormally on a transaction, while the other behaves normally (clerks do not collaborate), and external consistency can be achieved.

These implementation techniques can be combined to suit the application. For example, returning to the MAC implementation described above, we did not consider how the enterprise might establish a secret key between a supplier and the system. Suppose that a supervisor is given this responsibility. So long as the supervisor (infrastructure) and the cnote-processing clerk are different people, then a failure by one cannot result in an unexpected behaviour at the external interface. This is an example of combining MACs and segregation of duties.

4 Conclusion

By considering the nature of the entire enterprise we provide a meaningful and implementation independent definition for integrity. This systems view [10] has not been adopted by conventional integrity models, such as Clark-Wilson [3], which limit themselves to the boundary of the computer system and tend to define integrity in an operational/implementation-oriented sense.

The properties used for verifying authentication in [4], and fault-tolerance in [9,11], are similar, and are based on non-interference. Therefore, we think it reasonable to claim that our notion of external consistency captures the sentiment of what is sought when analysing authentication protocols in general[1]. However, our interest is not in verifying authentication protocols, per-se, but in verifying the external consistency of more general enterprises that use a variety of techniques to implement integrity. Our formal definition of external consistency is for illustrative purposes only. A re-formulation of this definition in terms of the existing verification approaches, such as [4,8,7,9], would be preferable. Additionally, given the usefulness of BAN [2] style analysis of protocols, it would be interesting to determine whether a similar approach could be used for analysing external consistency in general.

Finally, we argued that external consistency is effectively an information flow property. There is a vast literature studying the nature of these properties (both possabilistic and probabilistic), the results of which, must be interpreted carefully. For example, preservation by composition will depend on a number of factors: the computational model, the structure of the interfaces, and so forth.

Acknowledgement

This work was done while I was a member of the CCSR, on leave from University College, Cork. I'd like to express my gratitude to Stewart Lee for inviting me and thank him, and the members of the Cambridge Computer Security Group, for a most enjoyable visit. Thanks also to Kan Zhang for discussions that helped to solidify my ideas on external consistency. This work was supported, in part, by a basic research grant from Forbairt (Ireland).

References

1. W.R. Ashby. *An Introduction to Cybernetics.* Methuen, New York, 1964.
2. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. Technical Report Report number 39, Digital Systems Research Center, February 1989.
3. D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security models. In *Proceedings Symposium on Security and Privacy*, pages 184–194. IEEE Computer Society Press, April 1987.
4. R. Focardi, A. Ghelli, and R. Gorrieri. Using noninterference for the analysis of security protocols. In *Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.
5. S.N. Foley. Evaluating system integrity. April 1998. Submitted for publication.
6. C.A.R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.
7. L.C. Paulson. The inductive approach to verifying cryptographic protocols. In *Proceedings of the IEEE Computer Security Foundations Workshop*, 1997.
8. A.W. Roscoe. Using intensional specifications of security protocols. In *Proceedings of the IEEE Computer Security Foundations Workshop*, 1996.
9. A.C. Simpson. *Safety through Security.* PhD thesis, Oxford University, Computing Laboratory, 1996.
10. J.F. Palmer S.M. McMenamin. *Essential Systems Analysis.* Prentice Hall, 1984.
11. D. Weber. Specifications for fault-tolerance. Technical Report 19-3, Odyssey Research Associates, Ithaca, NY, 1988.

[1] At this point we do not consider confidentiality, although the authors of [4] have used non-interference to characterise confidentiality in authentication protocols.

External Consistency and the Verification of Security Protocols
(Transcript of Discussion)

Simon Foley

University of Cork

I have been wondering what is meant by system integrity and I came up with an interesting result, at least I think it's interesting. And there's a tie-in to security protocols in the answer and I get to that later on in the talk. If you look at conventional models of system integrity, we talk about things like Clark and Wilson and Biba, they're all implementation oriented. They say: this is how we achieve integrity, we achieve integrity using things like segregation of duties, well-formed transactions, audit, and so on. And they say these are good things to use in your systems. So if you were to take Clark and Wilson and try to evaluate a particular system configuration according to their model, all it's going to tell you is that your system uses good design principles. It's not going to tell you that it actually guarantees integrity. So it's not going to say that if you use a particular system configuration you can bypass integrity by some very obscure circuitous route which you hadn't anticipated, so Clark and Wilson and Biba are operational models, they say this is how we achieve integrity they don't say what integrity is.

So starting from that I said well, what is system integrity? And the route to it is by trying to understand what is meant by external consistency. And in Clark and Wilson they said external consistency is the idea that what's in the system reflects what's in the real world. So if I had ten widgets sitting on the shelf of my stock room, then my computer system says there are ten widgets in the inventory. And that's what they said, this is what external consistency is, and they say we use things like segregation of duty to achieve it. So they don't really say, they don't give a formal definition of what external consistency is, they give a general explanation of what it is and how to achieve it using things like segregation of duties, auditing, and so on.

Now what I'm going to do. Well, what I was looking at was, can I formalise what is meant by external consistency. And in this very short talk, I just have a very simple example which I'm going to use to illustrate what I mean by external consistency. I had the whole thing formalised but I'm not going to look at any formal definitions here.

If we go back to traditional systems analysis, and this is something that people still do when they do, say, an object analysis, you say I want to design a system, you say what are the requirements of this system. So here's my really simple system, it's a black box at the moment, the supplier supplies us with a C-note (that's a consignment note, not a $100 bill) saying, here are your goods, and this system is supposed to produce paper. That's what my requirements analysis would say. I'm saying nothing about computer systems, I'm saying nothing about the people who are going to have to run it, at the moment we just have a black box, and then what do we do.

So requirements analysis tells us what the system should do. We then decide how we're going to implement it, and in this particular example, I say, OK we have the computer system which inputs an invoice, validates the invoice, generates a payment, which it sends out to the supplier. Now the computer system can't exist in a vacuum, and it has to have a person to operate it. So we have the clerk who takes the consignment note, basically interprets that consignment note as an invoice, types the details into the system and payment is generated. So there's my implementation and I'm saying this is how my system is going to operate. You can see here that when we think about implementation, we start thinking about the infrastructure the system's going to run on, you know, if I design a particular system I have to add in lots of additional redundant components. I have to say, oh I need a backup and restore system, because date can be corrupted and I want some way of retrieving the original data just in case of a problem. So the infrastructure that is going to be used to support requirements is almost always going to fail us. So that's my view of what I would think of a system.

Given that, let's define what we mean by external consistency. And again, remember, this basically says that the information in the system should correspond to information in the real world. And in this case I'm saying, well really what I want is an externally consistent interface, between the external entity of the supplier and the entire enterprise, not just the computer system, but the people who are also using it. The idea being that whenever the supplier sends in a C-note, the supplier is expecting a payment to be generated. But of course we know we have a potentially faulty system here, which may lose the C-note, which might try to pretend there was a C note sent in, and so on. So we have to have a sufficiently reliable entire enterprise that we can say, well we generate payment from the C-note.

So external consistency then I would define as follows. Ignore the full defnition, what we're really saying here is, if the infrastructure fails the system will still behave as we'd expect. A simple example here would be a clerk sets up their own supply company, then generates a stream of invoices internally pretending that consignments have been made and as a result a series of payments are generated and this isn't externally consistent because no C-notes were originally sent in, and this is a very simple unrealistic example I guess, because if you'd have a lot more controls in reality. So the external consistency is achieved if the behaviour of the external interface resulting from an infrastructure failure is consistent with the behaviour under normal conditions.

Now looking at this very briefly what we're really saying is that under normal conditions the system is going to behave like P2. The infrastructure, which is the clerk, is expected to behave like P1: gets the C-note, types it in, gets the C-note, types an invoice. And then the external interface's view is through this E, it just sees sequences of C-notes and pays. OK we now take P1-bar, which represents the infrastructure failure, the clerk becomes dishonest, and in this case it's saying the clerk can behave any way that they want to, and we combine that with the system. Are we still going to end up with sequences of interactions which would be valid sequences of interactions.

Virgil Gligor: Simon, if this property, maybe I'm not understanding it, if this property is achieved, why do you need the clerk.

Reply: We need a clerk who is going to be the physical part of the system. There has to be some sort of an interface between the supplier and the computer system. Now if I could supply to you a computer system which worked perfectly, let's say using ESP, the supplier sits down in the office and thinks, I have just supplied something to you. The computer system picks up this thought wave and generates a payment. Now you could say that in this case we have a perfect infrastructure because there's no possibility for failure. But in reality we'll never have a system like that, we'll always have our system running on an infrastructure that can fail, and in this case it's the infrastructure's a clerk who can fail, who can try to cheat the system.

Roger Needham: I think the point is, if you achieve external consistency whatever the clerk does then you may as well shoot him.

Carl Ellison: It might become consistent by shutting down.

Reply: Well, no. You're in the goods-inwards section of your business and something arrives on the doorstep with a note attached to it, a consignment note, how else is that information going to be entered into your system?

Virgil Gligor: You can have the supplier enter it in directly.

Reply: OK that's fine, we don't have potential failure there.

Virgil Gligor: It seems that what you're doing is something else. It seems like you are trying to model some failures in some components of the system you are analysing, an enterprise, and it also seems that you equate that in some sense with integrity. And I will agree that resistance to failure is part of a larger definition of integrity. I think that's a very valid observation but it's not, definitely not, the same notion of integrity that, for example, Biba had in mind. Clark and Wilson, you may want to say, have this notion of integrity because they look at redundant actions and separation of duty, they motivate separation of duty because of this resistance to single failures of omission of permission, so in that sense you may have an intersection with Clark-Wilson, but not with Biba.

Reply: Yes, OK. I guess I use Biba as an example of being another model which was more operational, in that it says here's a way to achieve integrity, but it doesn't actually. Another parallel that I could use is if you think of confidentially. If you think of, say, Bell-LaPadula and if you think of Gougan-Massenges' non-interference definition, Bell-LaPadula was an implementation-oriented operational model: he said, here is how to achieve security.

Virgil Gligor: An approximation.

Reply: An approximation, yes. While with non-interference they try to say, well let's forget about the operational detail and say what is it that we're actually trying to get, what is meant by confidentiality.

Virgil Gligor: I understand what you are claiming now, I think.

Reply: I have some more examples which may be a bit clearer.

Michael Roe: One way in which the clerk can fail is by doing nothing, and if the system carries on working normally under that failure mode then you didn't need them in the first place. But you need to be a little be more careful about what's a failure and what's not.

Reply: Well, yes. I have just a form of pen-and-paper spec of this, and when the clerk behaves like this, they can do anything that they want, they can engage in any sequence of actions, now there's always the possibility that they can just literally deadlock and jam up the entire system. But at least we won't have spurious payments being generated. So it's like in this case the system lags seriously behind the real world, so it's like a fail-safe, and if it fails then you still don't lose any money.

That's that particular definition, but there have been, I would expect there are variations of the definition that would give you a definition of external consistency which would try and include liveness.

Bruce Christianson: How insistent are you on the assertion that the thing that you're checking consistency relative to is *the* real world? How far could I tempt you into saying that it's consistency with another different model of the system.

Reply: Yes, sure. I won't answer the question but I'll just make an observation, that yes you're quite right, how do we know this is the correct external interface to the outside world. What I've done, I haven't done it here, but what I did do is, I took a system like this and said, OK now this is the interface and its external interface is this. And I have some simple construction proof which says you can do that. And then this system was basically a simple program or a simple set of three programs running on top of an assured pipeline. And in that case, the assured pipeline was the trusted part, and the individual programs then could fail. And you could prove external consistency in that case, so you could reduce it down and down, and in the same way I guess you'd go further on with this.

Bruce Christianson: And then you're no worse off than the database people.

Virgil Gligor: Another view of what you are doing, I think, is that you are presenting a method that identifies a part of the system that will not be trusted.

Reply: Yes, I'm saying the system has been run on top of an infrastructure which can fail us.

Virgil Gligor: So you need to partition now what needs to be trusted from what does not.

Reply: Yes.

Roger Needham: Maybe it would be helpful if you didn't talk about the real world, but just the outside.

Reply: Yes. I'll show you the examples.

So the question is, how do we implement external consistency. So I've said what is meant by external consistency, this is what integrity is, now how do we implement, how do we achieve, integrity. Well the first way to do it is use fault-tolerance. Let's replicate the clerks. In this case we'd say this is a bizarre system but it's an example. We have $2n + 1$ replicated clerks, so let's say we have three clerks, each of which receives the C-note, each of them says, I have an invoice, or is supposed to say, I have an invoice. We then have a simple voting system built in to P2 which looks for a majority vote and if it gets a majority vote on an invoice produces the payment. And so it's a very simple example, I guess of Byzantine clerks. In the failure model, OK the P1-bar which is the model of clerks failing, you say anything up to N clerks can fail, which means instead of having their original behaviour which is take a C-note, process invoice, take a C-note, process invoice, up to N of them can take on arbitrary behaviour, meaning they can generate a stream of invoices even though they never had a C-note. And under that we see that it's clear we have external consistency.

Stewart Lee: Providing the voting mechanism works.

Reply: Yes, we're assuming that this works. But of course we can take the voting mechanism and we can treat that as another enterprise which has trusted parts and untrusted parts.

OK, the other interesting observation I saw with this was that my definition of external consistency is really I think ultimately a non- interference type property. It's saying that this part of the system cannot interfere with the generation of payment. But it's a special kind of non-interference because we're saying, OK the changes in this part of the system behaved properly, behaved badly. So we're constraining the way it can behave and that it won't produce bogus payments. So if you think of external consistency as being a non-interference style property, I think the point really being made there is there's lots of work done on that, and all I'm saying is there are probably other more suitable non-interference based definitions that we could use to characterise external consistency. But there was work done in 1989 by Weber which was, to use non-interference as a way of verifying fault-tolerance. So I thought that was kind of interesting, that came around to that. And then there was some more recent work done using a CSPFDR to verify fault-tolerance using a sort of non-interference type property. All I'm saying is, the definition of external consistency is reasonable because at least for fault-tolerance the way its working ties in with what's been done already.

So then when we come to segregation of duties as another way of implementing external consistency, if you think of fault-tolerance as being something that replicates an operation, then segregation of duty, what it does is, it partitions an operation among a number of different people and in this case my system takes a consignment note and an invoice, and when it gets the two of them it generates the payment. OK. And my infrastructure, which in this case is the clerks, one will verify the consignment, the other will process the invoice. Now for simplicity I just have them coming from the same C-note, but you can imagine that they could come separately. And our segregation of duty would say this has to be done by two separate people. And we could say well if this is the case, if they are separate people, and if one of those people fails when we describe our infrastructure-bar, the failure bar, then the system will be able to say, OK there's a problem on this particular invoice, don't generate the payment. So it's like this fail-safe, so we won't generate payment unless both of them agree. So segregation of duty is an implementation of external consistency.

Another example is cryptography. If we go back to the original example where you just had one clerk, and suppose the system and the supplier both share a secret key and the supplier attaches a MAC to the C-note which then the clerk just types in, the MAC exactly plus the other details, and the system can then validate the integrity and the authenticity of the invoice and decide whether or not to produce payment. Now in this case the P1-bar, the failing infrastructure, corresponds to all possible fraudulent interactions for a clerk. So the clerk normally just alternates from C-note to invoice, for failure we have to basically model P1 saying, what can I do to try and attack this system. Now in this case we have external consistency but it's interesting to see here that this model corresponds to the kind of models that seem to be used when people analyse cryptographic protocols or authentication protocols, like Larry Paulson's work.

What we have in this case, if you think of the system as here being the protocol, the rest of the enterprise or the infrastructure is the network all corresponding to the spy whose doing the attacking, and then the P1-bar which is like the `synth` of all messages, all possible Freudian interactions[1], so here we have this idea that (it doesn't matter whether it's a cryptographic based protocol or there's other simple protocols) the basic property that they're trying to achieve here is this thing called external consistency. Now what's in the system corresponds to what's out in the real world, sorry, what's outside. And then with this we can combine the two, we can say OK we have our C-note with the MAC attached, how do we set up an initial key exchange. Well our system generates a new key, say it is distributed to the supplier by the supervisor, and so long as this supervisor here is different to the clerk you'll also have external consistency. If it's the same person of course that means then the clerk can generate bogus transactions.

Mark Lomas: Or invoices.

Reply: So I guess the only conclusion I want to make is this is just this: when a designer claims that a system is fault-tolerant, that a protocol properly authenticates, that the system is secure against fraud, then I think what they're saying is that it's externally consistent. That what's in the system corresponds to what's outside the system.

Bill Harbison: No comments on that final statement? I'm still taking it in.

Carl Ellison: I too am still taking it in, but I spent a number of years designing fault-tolerant systems a while back, and one of our results was that if every component in the system is fault-tolerant then the entire system can be.

Reply: Can be, yes.

Carl Ellison: But if there is any non-fault-tolerant component then at least at that interface change can achieve inconsistency in base terms. Since your

[1] Actually I said fraudulent interactions, but let it pass.

external component of this system is not fault-tolerant, because the human supplier doesn't have replicas and the delivered physical goods are not replicated, then I'm not sure that we could ever achieve consistency. I haven't proved it, but I remember proving this when it was just computer processing.

Reply: Yes, sure. I think here I was using the clerks, humans, to try and simplify the example as much as possible. I have one other example which basically took this computer system, and in the example I had three transformations and three programs. One was this, which was to update the state, another one was to process the invoice and the third was to generate the payment. So you can imagine three separate programs or processes which run. Now what happens if one of these fails, if it becomes corrupted or there's a flaw in the program, and as a result it could start behaving in an arbitrary manner. You'd want to make sure, say if this fails, that it can't result in the generation of bogus payment. So if you built this on top of, say, an assured pipeline TCB which would force you to do this one first, then process an invoice, then generate a payment, in lock-step like that. And you have each of those programs executing in a protected domain. Then in that case you can say, well you have external consistency on that particular interface. You'd have your infrastructure, which is three programs, your modelling of the faulty infrastructure is, one of these programs can fail, you have your reliable component, which is your assured pipeline.

Carl Ellison: I'm remembering a little more of that work we did years ago. One of the things that we discovered was that if you have a non fault-tolerant component, in this case the supplier, assuming everything in the enterprise is made fault-tolerant, that you can achieve one of two kinds of unfairness. I think of a transaction, a trade, we consider the trade fair if it's atomic, and we cannot achieve atomicity if there's a non fault-tolerant component. But we can choose which kind of failure we'll permit and it appears you've chosen to withhold payment if there's a failure, and if that's your definition of success, fine. ¿From the point of view of the supplier that's not very successful.

Reply: Yes, I hear. The definition that I used there is based on non-interference like properties. There's a lot of non-interference properties out there, so I just chose the simplest one so that I could characterise what it was I was saying. And I also chose the simplest computational model which is a simple trace model. So with the two of those I can't characterise any sort of liveness properties. Things like, the supplier will get paid. But if you think of it in another way you could say, well what we could do is we could set it up so that every now and then there's a check between the enterprise and the supplier, or the supplier has the ability to phone up and say where's my payment, and as a result from that we'd expect some payment generated before we process another consignment. Ultimately if you're going to talk about liveness you'd have to use a different model.

Bill Harbison: I think you covered yourself though in your statement Simon, because your statement talked about the designer. And that may very well be a true statement for the designer, it certainly may not be a true statement for either the user or the implementor of the system and there is more than one party involved which is I think what we're saying here, in all of this. And this highlights the fact that just saying the design is correct doesn't mean to say the system works according to the way that people expect it to work.

Reply: I think if I was to motivate again why I looked at this, it was when I was looking at integrity. I said, well what is integrity, what does it really mean. And the problem I could see was, all that people do is say oh, here's how to achieve integrity: use MACs, use segregation of duties, etc. And I said, well if I have a very, very large complex system with a lot of operation details, a banking system, with proper segregation of duty rules set up, and everything else, how do I know that it's not possible for a clerk, how do I know that there's not some funny route that they could take which would end up with them being able to defraud me.

Audience: You don't. When you were talking there about the possibility of the supplier phoning up and provoking something, that opens all kinds of possibilities of fraud.

Reply: Sure, but one would presume that you would include those kinds of details in the component analysis.

Stewart Lee: How do you know that you're not.

Reply: It's a closed system.

Virgil Gligor: There are two things here that ought to be separated. One is the policy that you claim here, that deals with certain processes of attack that are related to the behaviour of these external people. And then there is another set of properties, which are penetration-resistance properties of the system, which are totally separable from the policy properties. Right? I can have a system that has penetration resistance, in terms of property X, Y and W, and I can change the policies underneath any way I want. So when you make those claims that you're trying to make, I think the first step would be to separate the two concerns, the policy versus the penetration resistance. Then you can reason about these two things separately which helps quite a lot.

Reply: You would build a policy on top of this, because all this is saying is that here is a widget that you can interact with through some interface. The hard part is to say, well what are the interfaces in this case. You build your policy up in terms of those interfaces.

Virgil Gligor: But in terms of the interfaces being secure, in some sense you are really talking about penetration resistance properties, which are completely separable once again from policy properties.

Bruce Christianson: But the essential move that Simon's making here is to say, we have some sort of specification of a system, in a world in which there's no fraud, no faults, only green fields with cuddly animals, that has the correct operational banking specification, and the question is whether this alternative specification is a correct refinement into an environment where there are these bad things.

Virgil Gligor: If he's doing that then he's actually focussing on policy and he's assuming that the system underneath, this green pasture, means that the system is penetration resistant.

The Trust Shell Game
(Position Paper)

Carl Ellison

Cybercash, Inc., cme@cybercash.com or cme@acm.org.

Abstract. Public key certification, network security and electronic commerce are all tightly bound to the concept of "trust". Nearly every paper deals with the word. Yet, few if any define it and none really defines it in a way useful to cryptographic engineers. Some uses of the word act like a verbal shell game, confusing the reader. These might be intentional or innocent: a product of trying too hard to convince the reader or just careless use of a word thought understood by all. As an alternative, this paper proposes a more concrete approach to the body of issues falling under the label of "trust" – one meaningful to cryptographic, security and computer engineers.

1 Introduction

One can not read much about security on the Internet without encountering the word "trust", almost always without any qualifiers. Currency of the United States carries the words: "In God We Trust". With God it is safe to use the word "trust" without qualifiers. With humans, that rarely makes sense. Therein lies the possibility for multiple shell games.

2 Shell Games

Shell games involving "trust" come in a variety of styles. Some are blatant and some are subtle. There is no evidence that any is intentional, but some might be. In any case, the reader should beware.

Some of these shell games rely on traditional logic errors. For example, the false implications

$$(A \Rightarrow B) \Rightarrow ((\neg A) \Rightarrow (\neg B))$$

and

$$(A \Rightarrow B) \Rightarrow (B \Rightarrow A)$$

show up quite frequently.

Some of these simply rely on the idea of "trust" as a magic salve. That is, once trust has been applied to some entity, the reader or customer is expected to feel warm and fuzzy about that entity, no matter how narrow the focus of that established trust.

2.1 Espionage Agency

Perhaps the most blatant trust shell game is played by intelligence agencies in order to gain access to cryptographic keys. The logic of this game starts with the false assertion that you need some trusted third party in order to establish identity[4] on the Internet.

> "Without a KMI of trusted certificate authorities, users cannot know with whom they are dealing on the network...."[8]

The logic then states that loss of a private key could be very troublesome, leading to a need to get new certificates, cancel old ones, etc. One therefore needs a backup mechanism for a private key and that mechanism must be trusted, because a private key is so sensitive.

There is only one entity with trust established so far in this conversation, a CA, so that is clearly the agency needed to perform private key backup.[5] This puts a private key where a government agency can demand its release.[2]

2.2 CA Salesman

A number of companies have sprung up to sell (name,key) certificates. The phrase "trusted certificate authority" is frequently found in the literature to describe a CA whose cryptography is trusted. The CA salesman cites that literature and then goes on to claim that because his CA is trusted, you can trust the certificates it issues. He then says that you need to trust the other party's key in order to do anything with it, so the other party needs a certificate from the CA.

2.3 CA Lobbyist

The logic of the CA lobbyist is a little more involved. The goal of this lobbyist is to make a market for his product and at the same time prevent competition as much as possible.

The thread goes roughly like this: "People need to know with whom they're doing business on the net. We are a trusted CA. We offer identity certificates. Therefore, we offer what people need. You should mandate certificates from organizations like ours. You should also not compete with private business, therefore you should legislate that the government not issue certificates. Meanwhile, you should protect us from liability, in order to encourage us to perform this socially valuable function. You should also make a keyholder completely liable for things digitally signed, because digital signatures offer non-repudiation."[10]

2.4 Cryptographic Researcher

Many research papers in cryptography, especially dealing with certification, use the word "trust" as a mathematician would use any variable in an equation. In this case, “trust” is the thing a certificate transmits. Its specification is deferred for the reader. The researcher then goes on to study the breaking of this unspecified trust. One does not need to know what the trust is in order to observe its having been broken by flaws in cryptography.

Unfortunately, this leads to the habit of claiming that if the cryptography and protocols behind issuance of a certificate are strong enough, then one can trust the certificate or, worse, that one can trust the certified key or, worse, that one can trust the keyholder.

3 Background to the Shell Games

3.1 Identity

Diffie and Hellman in their seminal paper on public key cryptography[3] said that with a (name,addr,key) in a phone book analogue you can do without trusted couriers.

Kohnfelder elaborated on this in his bachelor’s thesis[7], defining the word **certificate** to mean a (name,key) entry from Diffie and Hellman’s phone book, but digitally signed and detached from any directory.

These papers assumed that a name functions as an identifier of persons and that what one needs is to identify the keyholder. The (name,key) certificate has since been labeled an *identity certificate* and people use that term as if it were a correct label.

The chain of logic here is: name → person → characteristics → identity.

“characteristics → identity” is valid by dictionary definition.

“person → characteristics” is valid in a community small enough that the receiver of the identifier knows the person being identified.

“name → person” is valid in a community small enough that the names people use for one another are unambiguous.

The trouble here is that the Internet is not a small community but victims of the shell games may not have realized all of the implications of that fact. For example, a CA issuing an “identity certificate” attaches its choice of name to the subject’s key, but leaves the certificate user to guess which CA-generated name applies to the person the user has in mind. Wherever there is a human guess involved, there is an opening for security mistakes and attacks.

3.2 Non-repudiation

Non-repudiation would be desirable and some cryptographic researchers worked to remove flaws in certification that would prevent non-repudiation. However, non-repudiation depends on establishing that a private key was used by a particular person to digitally sign a particular document. This requires protection of the private key from theft, protection of all software driving use of that key to prevent signing of unintended documents and protection of the computer system from use by unauthorized persons. The technology to achieve the first two is not available today and the last is extremely expensive, so non-repudiation is probably not possible today.

If a researcher assumes those protections exist, non-repudiation can be established. Papers from such researchers are then cited by CA lobbyists to get laws passed declaring keyholders responsible for any action of their private keys.[6]

3.3 Electronic Commerce

The advocates of identity certification advance the notion that what you need to know in order to do business is who the various parties are, with the unstated assumption that if you know the names of people, you know who they are.

Two people who want to do business need to trust one another in very specific ways. The merchant needs to trust that the consumer will pay. The consumer needs to trust that the merchant will deliver the purchased goods. These are characteristics of keyholders, not of their names, and thus the identity flaws described above apply.

4 An Engineer's Alternative to Trust Shell Games

If we look at security from the point of view of the verifying program, that program needs to make a decision, typically of whether to allow access to some resource (a computer system, files, money, ...). That program deals with keys, which "speak for"[1] their keyholders, and needs to answer, "does this key have permission to access the resource it is asking to access?" As long as that permission is specified explicitly and its delegation is well controlled, there is no room for the shell games described earlier. One mechanism for well-controlled granting and delegating of permissions is summarized in Appendix A.

References

1. Abadi, Burrows, Lampson and Plotkin, "A Calculus for Access Control in Distributed Systems", DEC SRC-070, revised August 28, 1991.
2. Denning, Dorothy E., "Response to the NRC report", June 1996, `http://guru.cosc.georgetown.edu/~denning/crypto/NRC.txt`
3. Diffie and Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, November 1976, pp. 644-654
4. Ellison, Carl M., "Establishing Identity Without Certification Authorities", 6th USENIX Security Symposium, July 1996, `http://www.clark.net/pub/cme/usenix.html`
5. Ellison, Carl M., "Emergency Key Recovery Without Third Parties", `http://www.clark.net/pub/cme/html/rump96.html`
6. Kaner, Cem, "The Insecurity of the Digital Signature", September, 1997, `http://www.badsoftware.com/digsig.htm`
7. Kohnfelder, Loren M., "Towards a Practical Public-key Cryptosystem", May 1978.

8. McConnell and Appel, “Enabling Privacy, Commerce, Security and Public Safety in the Global Information Infrastructure”, report of the Interagency Working Group on Cryptography Policy, May 12, 1996; [quote from paragraph 5 of the Introduction]
9. SPKI documents: `http://www.clark.net/pub/cme/html/spki.html`
10. Winn, Jane K., Couriers Without Luggage: Negotiable Instruments and Digital Signatures, 49 S.C. L. Rev. (forthcoming May 1998).

A SPKI Authorization Control

SPKI[9] defines a certificate as a signed 5-tuple: $[I, S, D, A, V]$ – where I is the issuer (granter of authority), S is the subject (recipient of authority), D is permission to delegate, A is the authority being granted and V is a set of validity conditions (e.g., a date range and/or on-line tests). I is a key or the hash of a key. S is a key, hash of a key, SDSI name (to be reduced to a key or set of keys) or a K-of-N threshold of sub-subjects. [The definition is recursive, although we expect few people to use that kind of complexity.] D is a boolean. A is an S-expression – a list of byte strings or sub-expressions, but always starting with a byte string – defined by the application developer to express the kind of permission with which the verifying program must deal. For example, the S-expression `(tag (telnet cybercash.com cme))` might be used to grant permission for the subject to use `telnet` to enter the machine `cybercash.com` as user `cme`.

5-tuples can be reduced by the rule:

$$[I_1, S_1, D_1, A_1, V_1] + [I_2, S_2, D_2, A_2, V_2] = [I_1, S_2, D_2, A_1 \circ A_2, V_1 \cap V_2]$$

if $S_1 = I_2$, D_1 is TRUE, and the two intersections (of A and V) are non-empty. $V_1 \cap V_2$ is normal date range intersection (after any on-line tests have been performed to yield date ranges) and $A_1 \circ A_2$ is authorization intersection (most often in which one S-expression is an initial sublist of the other and the longer one is the result).

SPKI defines an access control list entry as a 5-tuple, not signed, in which I is the reserved word `self`.

When the verifying program runs, it must produce a 5-tuple reduction result of the form $[\texttt{self}, S, -, A, V]$ where S is the key signing the access request, “$-$” indicates that we don’t care if S is allowed to delegate, $A \circ A_0 = A_0$ where A_0 is the access request, and V includes the time of the request.

The Trust Shell Game
(Transcript of Discussion)

Carl Ellison

Cybercash

I wanted to hammer on one point that Joan made, brought up in Larry's talk this morning and I borrowed a slide from him. When he was talking about verifying the certificate, assuming the certificate is verified correctly, then he has established that the client is present, and I think he was quite right not to delve into the process of verifying certificate change. That's gotten needlessly complex, in fact I personally believe it's a case of taking a simple job and assigning too many people to it until you end up with something that is blown up entirely out of proportion.

The problem I have and I don't mean to pick on Larry, but I couldn't resist this opportunity. The problem I have here is we frequently talk like this, we say if the certificate chain was verified correctly then this is true. Or we get into that other camp of folks who just want to talk about certificate chains. Now what nobody asks or talks about is what does this mean. Just as Joan did not want to define the word trust. We use the word trust all the time which is what my written paper is about and I really am not going to cover much of that, I'll let you read that. We use the word trust all the time without defining it. I have an example out of my past that brought home the word trust for me.

There was a woman knew, some number of years ago, and her husband always used their automobile and so she was left without an automobile and she was a friend of mine so I ended up driving her around places and the more we'd drive the more we'd talk and we got to be pretty good friends. Well one day after about a year of this I ran into her husband when I was driving her home. I met her husband and I said doesn't it bother you that I keep driving your wife round all the time. And he said no it doesn't bother me, I trust her. Perfectly valid, legitimate statement for him to make. I had gotten to know her of course and as a personal friend I trusted her. So I too would make the statement I trust her. I happen to know about all the affairs she was having and he didn't, so clearly we had two different definitions of the word trust. But we could both use the statement I trust her.

Well ever since that, I have been wary of using the word trust and especially using it without quantifying it and that's mostly what I was talking about in that printed paper. So instead of using the word trust I use, in the SPKI work, the word authorisation. I should mention that I'm sitting between two talks of AT&T work, what I'm talking about is not AT&T work, it's just that we happen to share a number of concepts. So I'm going to talk about some forms of certificate and in particular about the SPKI work, which is an Internet engineering task force standards activity. Let me start by mentioning there are many classes of certificate. We keep using the word as if it meant only this first client certificate mapping a name to a key, and we call that an ID certificate, or identity certificate, frequently. This was the certificate that was defined original, this was the concept that Diffie and Hellman brought up in their paper. They said we can get rid of these secret key couriers by publishing a telephone book, the telephone book will list name, address and public key. It's a great idea provided that a name means something, the assumption here is that a name is a valid identifier. Actually, there's two assumptions.

The first one is that the name is a valid identifier and the second is that once you've identified someone that's all you need to know. No Joan has hammered on that second point some. Let me address the first one. A name is a valid identifier provided there are two conditions for it. One is that the name is unique, that it uniquely identifies, uniquely labels, some individual and the second condition is that any user of that name needs to know who that name refers to. When we use common names in our normal lives these names are valid as identifiers. If I talk to Joan right here about Matt we both know who I'm talking about. I know that she knows that I know who I'm talking about and so forth. But once you get a large enough community this is no longer true. And in particular in the Internet commerce community, this is definitely not true.

Merchants want to have customers come to them that they have never met, customers from all over the world. So first of all names, common names, are not going to be unique, they have to be extended, and secondly the extension is not necessarily known to the person who is using the name. This leads to a protocol failure, or can lead to one. Let me mention it in a second. Another kind of certificate that we may have seen described is an attribute certificate. The strongest description I've seen of this is in X..57. Attribute certificate maps are commissioned to a name and the assumption is that with a pair of these you can get from a specific permission to a key. Let me give an example. I know of a bank that wants to issue attribute certificates for accountants that would give them permission to do electronic banking on-line and so let's say I go down to my bank and I say I want a certificate to give me permission to do electronic banking.

Let's start back up here, let's say I want to do some electronic banking and all I've got is this TLS certificate for SSL, client certificate listing my name, and attaching my name to a key. The CA that assigned me that certificate made it unique, made my name unique by adding information to the name Carl Ellison. I don't know exactly what information they would add, but they'll add something, so that now this full name, this full distinguished name, is unique. I present this certificate to the bank's web page by SSL/TLS and the bank can now see this connection is coming from Carl Ellison number 423736 or whatever extra information was added, and that person is trying to get access to this account which belongs to Carl Ellison. The temptation is to match those two names and say OK let him have access. What they can match of those two names however is a subset of the distinguished name and a subset of their information on file and that policy that I just described is saying if there is any intersection between identity information of file and identity information in the certificate, give the person access. What that means is that access has just been opened up to hundreds of people to my bank account. Needless to say I don't want that to happen.

The same potential flaw in a protocol happens if you use these two certificates in conjunction with each other. The assumption here is that you'll get a name to a key certificate from some commercial CA and the permission to name certificates from your bank. That has the same potential flaw that the website had except it happens once when this certificate was issued, rather than each different attempted access. So anyone named Carl Ellison could go up to their bank and get this certificate and then from then on have what looks like very solidly credentialled access to my bank account. Needless to say this is a problem. What I've been advocating, what Joan advocates, what Angelos advocates are permission key certificates, authorisations for a particular key, avoiding names in this process, although there are plenty of times when a name is valid, for electronic mail I use nicknames for people. My nickname for Matt Blaze is MAB. It happens to be his initials but it's what I use as my name for him and so I might want a certificate that I issue binding my name MAB to Matt's public key. So I might in fact generate a certificate like that. It's just that we're not talking about local name certificates here.

Michael Roe: Matt's .... not being used like other certificates, because a certificate from somebody else binding their MAB to a different key would be of no use to you and it's your key pointing MAB to that key is no use to anybody else.

Reply: Unless that certificate from the other person was Joan's because then I know that Joan knows Matt by the same initials that I know Matt, I know that what we're using is an intersection in our name spaces where they agree.

Bill Harbison: How would you formalise that?

Reply: I have no idea.

Joan Feigenbaum: SDSI

Reply: Well SDSI is one way to formalise it, but it doesn't formalise the intersection the way I just described it but it is a formalisation of this intersection.

Virgil Gligor: I don't quite understand something about your permissions. Permissions have to be unique which means that usually if I'm allotted to some type which comes perhaps from a type hierarchy that has a unique root in the system on and on and on, so what I'm trying to hint by this semi-question is that I don't see how permissions are different from names.

Reply: As Rivest and Lampson showed in their SDSI paper, you can define names for any kind of permission that you want to issue.

Virgil Gligor: If I do that I can do that for names.

Peter Landrock: It's just a question of distinguishing names, isn't it.

Michael Roe: It's a matter of what you're naming, if you're naming permissions you're naming something within the electronic information processing system and you're on good ground. If you're naming people from the outside world, you're taking this route outside into external consistency and back in again.

Virgil Gligor: I can move them into my realm of partners.

Reply: I don't think the issue is, in an SPKI certificate there is a field which we call the tag which gives this permission and you can think of this tag as a name if you want to. The key issue is who is allowed to issue that certificate.

Virgil Gligor: I still don't see it. I think that Michael has a closer answer to that by saying that there is a dichotomy between some value that's a known a priority internal in the system. There's some common knowledge about this, and common knowledge is hard to achieve I know that, but then some common knowledge about permission and that simplifies the problem because names they present identities of unknown entities, so that I would buy partly as an explanation, but so far I don't see how SDSI solves anything in the dichotomy of names and permissions.

Reply: We do not use in SDSI or in the original SPKI or in PolicyMaker or in KeyNote, we do not use any globally defined permissions. Period. Instead if you think of, I don't have my physical keys with me, but you think of a brass door key or my ATM card. This is, I'd like to cite this as an authorisation certificate, because this ATM card has no name on it, you know this is not an example of a name certificate which oh by the way gives me access to money, this is a tool for getting access to money. It is not an identity certificate at all.

Virgil Gligor: It's a capability.

Reply: It's strictly capability. But it's not just a capability because you need more than possession of this.

Stewart Lee: That's right, but it's got your name on it, it's just that you don't have the right kind of eyes to read it. You need magnetic eyes to read it.

Reply: It happens to have my name on the magnetic stripe, it doesn't need that in order to function to give me money. I could erase my name from the magnetic stripe and still get money.

Peter Landrock: But it still has an identification and whether that is directly connected to you.

Reply: If course it does.

Michael Roe: The door keys are a better example ...

Reply: It identifies the bank account from which I can get money.

Bill Harbison: You have shifted the responsibility from who exercized the task to who authorised the task to be exercized.

Overview of the AT&T Labs Trust-Management Project
(Position Paper)

Joan Feigenbaum

AT&T Labs – Research
180 Park Avenue, Room C203
Florham Park, NJ 07932-0971 USA
jf@research.att.com

1 Introduction

Emerging electronic commerce services that use public-key cryptography on a mass-market scale require sophisticated mechanisms for managing trust. For example, any service that receives a signed request for action is forced to answer the central question "Is the key used to sign this request authorized to take this action?" In some services, this question reduces to "Does this key belong to this person?" In others, the authorization question is more complicated, and resolving it requires techniques for formulating security policies and security credentials, determining whether particular sets of credentials satisfy the relevant policies, and deferring trust to third parties.

Since the Autumn of 1995, a small group at AT&T Labs has conducted a broad research project in Trust Management. Our main focus has been the design, implementation, and use of the PolicyMaker trust-management system [4,6]. This paper presents some of the findings and open questions that our group has produced thus far. Reasoned controversy may be stimulated by some of the following design choices:

- **Authorization not authentication**: In answering the question "is the key used to sign this request authorized to take this action?," a trust-management system should not necessarily have to answer the question "whose key it is?" Rather, the trust-management framework should support credentials that directly authorize keys for certain types of actions.
- **General compliance-checking mechanism**: The mechanism for checking that a set of credentials proves that a requested action complies with local policy should not depend on the semantics of the application-specific request, credentials, or policy. Because designing, implementing, and proving the correctness of a compliance checker is a non-trivial task, it should not have to be done from scratch for each application.
- **Programmability**: Policies, credentials, and trust relationships should be expressed in a simple but general programming language. The language should be expressive enough to support the complex trust relationships that

can occur in the very large-scale network applications currently being developed. At the same time, simple and standard policies, credentials, and relationships should be expressible succinctly and comprehensibly.
– **Locality of control**: Each party in the network should form the "trust root" for its own environment. That is, local policy should determine in each transaction whether to accept the credentials presented by a second party or, alternatively, which third party should be asked for additional credentials. Local control of trust relationships eliminates the need for the assumption of a globally known, monolithic hierarchy of "certifying authorities."

2 Discussion

We will now discuss each of these decisions briefly and give some of the reasons that they are controversial.

2.1 Authorization not Authentication

Traditional "public-key certificates" that bind identities to public keys have received a lot of attention (*e.g.*, [1,2]). Acceptance of the goal of reliable, large-scale creation, distribution, interpretation, and, when necessary, revocation of such certificates is based on the assumption that the way every product or service should deal with a digitally signed request for action and a public key that can be used to verify the signature cryptographically is to ask "who owns the private key that corresponds to this public key?" and "is the owner of that private key authorized to take this action?" We reject this assumption and hence reject the goal.

Why would a reliable answer to the question "who owns the private key that corresponds to this public key?" allow a product or service to answer the question "is the owner of that private key authorized to take this action?" In mass-market scale applications, the identities of signers usually won't be known to the receivers of signed requests. Thus they cannot appear on receivers' access-control lists and other identity-based policies for deciding who can do what. Our position is that, rather than supporting signed requests with name-key binding "certificates," requesters should support them with more flexible and general "credentials" that prove that keys are authorized to take actions.

Note that a reliable system for associating a persistent "identity" with a public key would provide *accountability after the fact* for any actions taken in response to signed requests. Developers and users should thus ask what they are really trying to accomplish by processing digital signatures. If it is technically feasible and cost-effective to track down people whose signatures resulted in objectionable actions, then traditional certificates would be useful if they could be deployed in a reliable way. In the many contexts in which after-the-fact accounting is either technically infeasible or cost-ineffective, digital signatures are only useful if they are supported by credentials that *prove before the fact* that the signer is authorized to take the requested action.

2.2 General Compliance-Checking Mechanism

An essential part of the goal of our "trust-management" project is to build a *general-purpose, application-independent* system that processes queries of the form "does request r, supported by credential set C, comply with policy P?" Why is this a reasonable goal? Since any product or service that requires some form of proof that requested transactions comply with policies could use a special-purpose compliance checker implemented from scratch, what do developers, administrators, and users gain by using a general-purpose compliance checker?

The most important gain is in soundness and reliability of both the definition and the implementation of "proof of compliance." Developers who set out to implement a "simple," special-purpose compliance checker (in order to avoid what they think are the overly "complicated" syntax and semantics of a general-purpose system like PolicyMaker) often discover that they have underestimated their application's need for proof and expressiveness; as they discover the full extent of their requirements, they may ultimately wind up reinventing the wheel and implementing a system that is as general and expressive as the "complicated" one they set out to avoid. A general-purpose compliance checker can be explained, formalized, proven correct, and implemented in a standard package, and applications that use it can be assured that the answer returned for any given input (r, C, P) depends only on the input and *not* on any implicit policy decisions (or bugs) in the design or implementation of the compliance checker.

Having made the decision to use a general-purpose compliance checker, we still have to choose and defend a notion of "proof of compliance." A detailed definition and analysis of the proof system currently used in PolicyMaker is given in [6].

2.3 Programmability

The two kinds of PolicyMaker *assertions* are *policies* and *credentials*. Both kinds of assertions have the same form, with two important components: a *source* (the keyword POLICY in one case and a string representing a credential-issuer in the other) and a program that specifies the nature of the authority that the source is conferring as well as whom or what it is conferring authority on. (See [4,6] for details.) These can be general programs, written in any language for which a "safe" interpreter can be incorporated into the trust-management environment. So far, we have used AWK, only because the pattern-matching features are useful for the policies and credentials we have experimented with and because a "safe" version was conveniently available. The question of which fully expressive programming language or languages are most suitable for policies and credentials is interesting and still wide open; the controversial decision we have made so far is to use general programs.

The most common arguments against full programmability of assertions are that it makes assertions incomprehensible and that it is not necessary. We have only the most straightforward response to the incomprehensibility argument:

While it is certainly possible to write incomprehensible assertions using a general programming language, it is not inevitable that this will be done, and indeed we offer the examples in [5,7,9,10] as evidence that programmability and comprehensibility can co-exist; in fact, we believe that users who need to express complex, real-world policies and credentials in restricted languages will be more likely to produce something incomprehensible than they would have been if given a more expressive language.

Our response to the claim that full programmability of assertions is not necessary is that repeated attempts to design adequate, partially programmable assertions have failed. For example, the Simple Public Key Infrastructure (SPKI) proposal [8], like PolicyMaker, supports direct authorization of public keys but rejects the notion that such authorizations should be fully programmable; over time, the original set of authorizations that could be conferred by SPKI credentials has been enlarged and changed several times, in response to what we believe are inevitable needs of applications that cannot be foreseen when any fixed set of authorizations is chosen. Very recently, we designed KeyNote [3] in an attempt to provide precisely enough generality and programmability to serve the trust-management needs of a Public-Key Infrastructure, if not the trust-management needs of a broader class of applications. It is close enough to being a "subset of PolicyMaker" to permit KeyNote requests and assertions to be processed by the PolicyMaker compliance-checker almost unchanged. I predict that there will be reasonable PKI demands that KeyNote's restricted assertion structure cannot accommodate and that we will wind up responding to such demands by saying "use PolicyMaker."

2.4 Locality of Control

The idea that explicit, local policy should form the "trust root" for decisions about all potentially dangerous actions has provoked less controversy than our other design decisions (perhaps because this approach has already been taken in certain special cases, including the popular PGP [13]). There may eventually be considerable controversy about how to implement this decision. "Local control" sounds very good in the abstract, but it is most straightforwardly done in a way that requires considerable configuration management by administrators and users. People who embrace the abstract idea that they should control their own "trust relationships" may resent the increased responsibility to choose and maintain security policies. Balancing the demands of local control on the one hand and ease of use and administration on the other is an important upcoming challenge.

3 Open Questions

Of course the fundamental open questions are whether any general-purpose "trust-management system" will ultimately be widely used in diverse applications and, if so, which one(s). Other open or partially open questions include:

1. Which programming language(s) should be used for policies and credentials?
2. Application areas in which general-purpose trust-management systems have already been deployed include web-page labeling [5,7], signed email [10], and copyright management [9]. Are these the most effective test beds for the idea of general-purpose trust management? Which other application areas should be explored? What evidence of the success or failure of the idea could the developers, users, and administrators of these applications provide?
3. What are some important and challenging elements of security policies that a good trust-management system should facilitate for a wide variety of applications?
4. What is the right way to define "credentials proving that a request complies with a policy"? Is the PolicyMaker approach of defining and analyzing a proof system from scratch reasonable, or would it make more sense to build a trust-management module on top of an existing logic-based programming language or other "inference engine" with standard formal semantics?
5. Where should the boundaries be drawn between a trust-management system and the applications that use it? For example, should credential-fetching and digital signature verification be the responsibility of the trust-management system or the calling application?

Acknowledgments

AT&T participants in the trust-management project include Matt Blaze, myself, Jack Lacy, Dave Maher, and Martin Strauss. Some of the work is joint with Yang-hua Chu (CMU), Angelos Keromytis (Univ. Pennsylvania), Brian LaMacchia (Microsoft), and Paul Resnick (Univ. Michigan). This position paper is my version of the results (to date) of several years of extensive group discussion.

References

1. *Information Technology – Open Systems Interconnection – The Directory:Authentication Framework*, Recommendation X.509, ISO/IEC 9594-8.
2. International Telegraph and Telephone Consultative Committee (CCITT). *The Directory – Authentication Framework, Recommendation X.509* 1993 update.
3. M. Blaze, J. Feigenbaum, and A. Keromytis, *The KeyNote Trust Management System*, work in progress. Internet Draft, March 1998, `http://www.cis.upenn.edu/~angelos/draft-angelos-spki-keynote.txt.gz`.
4. M. Blaze, J. Feigenbaum, and J. Lacy, *Decentralized Trust Management*, in Proceedings of the Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos, 1996, pp. 164–173.
5. M. Blaze, J. Feigenbaum, P. Resnick, and M. Strauss, *Managing Trust in an Information-Labeling System*, European Transactions on Telecommunications, 8 (1997), pp. 491–501. (Special issue of selected papers from the 1996 Amalfi Conference on Secure Communication in Networks.)

6. M. Blaze, J. Feigenbaum, and M. Strauss, *Compliance-Checking in the Policy-Maker Trust Management System*, in Proceedings of the 2nd Financial Cryptography Conference, Lecture Notes in Computer Science, Springer, Berlin, 1998, to appear. Available in preprint form as AT&T Technical Report 98.3.2, `http://www.research.att.com/library/trs/TRs/98/98.3/98.3.2.body.ps`.
7. Y.-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss, REFEREE: *Trust Management for Web Applications*, World Wide Web Journal, 2 (1997), pp. 127–139. (Reprinted from Proceedings of the 6th International World Wide Web Conference, World Wide Web Consortium, Cambridge, 1997, pp. 227–238.)
8. C. Ellison, *A Simple Public-Key Infrastructure*, `http://www.clark.net/pub/cme/html/spki.html`.
9. J. Lacy, D. P. Maher, and J. H. Snyder, *Music on the Internet and the Intellectual Property Protection Problem*, in Proceedings of the International Symposium on Industrial Electronics, IEEE Press, New York, 1997, pp. SS77–83.
10. R. Levien, L. McCarthy, and M. Blaze, *Transparent Internet E-mail Security*, `http://www.cs.umass.edu/~lmccarth/crypto/papers/email.ps`
11. P. Resnick and J. Miller, *PICS: Internet Access Controls Without Censorship*, Communications of the ACM, October 1996, pp. 87–93.
12. R. Rivest and B. Lampson, *SDSI: A Simple Distributed Security Infrastructure*, `http://theory.lcs.mit.edu/~rivest/sdsi11.html`.
13. P. Zimmermann, **PGP User's Guide**, MIT Press, Cambridge, 1994.

Overview of the AT&T Labs Trust-Management Project
(Transcript of Discussion)

Joan Feigenbaum

AT&T

A number of people have started talking today by saying I wanted to figure out what such and such term means. What does integrity mean? What does confidentiality mean? I definitely don't want to give you the impression I'm going to tell you what trust means. Trust management is supposed to be an incredibly vague and provocative term invented by Matt Blaze. I don't know whether he intended it that way, but it comes naturally to him. What I find is that trust management the term induces this projection. People hear it and they think "Oh, that sounds good, that's what I need". So that's good. So what I am going to do is give you a motivating example, tell you a little bit about what we're trying to accomplish by building a trust management system and what are the essential design decisions we need when we build the PolicyMaker trust management system and then some open questions.

Now, what I probably won't do because of time and in particular because the three of us want to leave time for discussion is something that is a whole separate talk which is to explain the algorithmic guts of the policymaker trust management system, but I have an old paper on that which I presented at the Financial Crypto Conference in Anguila in February.

So the motivating example, I think this is a straw man example, I'm not sure, we'll see who believes that. Suppose somebody is trying to set up a business, a programmer named Bob trying to set up a business. The idea of how the digital signatures on the Internet could facilitate lots of little small businesses along these lines is that Bob, this merchant, could write the software, sign it, and post it on his web site. Why is he signing it? Well he's trying to make sure everybody knows this is his so that his good customers can spread the word about him and he can build up the business. So a wary customer named Alice should know about the potential pitfalls of, essentially what can she do, she can verify a signature but she still has to convince herself she's using the right verification key. So Alice, if she's going to do this convincing of herself using traditional certificate mechanisms would do what? She'd go to some directory and look up Bob's public key, and she'd look up a certificate for Bob. She would verify that this really is Bob's certificate using a well known public key of a certifying authority, then she'd verify the signature on the software using Bob's public key which she has now convinced herself really is Bob's public key and now she'd know that it really was Bob who signed PK and she could go ahead and be his customer. Well that's sort of the way traditional say X.509 certificate goop would have you do it. Maybe you wouldn't get one certificate from one certifying authority which means you'd have to have a chain of certificates but that's the idea. Trying to verify the name key binding. Bob and PK_{Bob}.

So I guess this is our starting point in the sense that we think that. Even if you could do that correctly. In fact has been a huge amount of systems effort, a huge amount of standardisation effort, and a huge amount of ink spilled about how do you create and distribute and crypt this certificate infrastructure that would allow you to do exactly that. It's not trivial to do that but what we're saying is this isn't what you really want to do. Because for Alice to know that it was Bob who signed this piece of software won't really tell Alice what we think she wants. In general, in an Internet style e-commerce world, at least the world that's been sold to us, the world we're imagining, when you get a signed message or you download a signed object from somebody somewhere knowing who's name is associated with the verification key is not really going to help you. In general this won't be somebody you know.

So there is a situation where Bob would not be Bob, this small businessman in his garage. Bob would be Microsoft or some other large software vendor, so if you're really just trying to figure out if that this actually was signed at the Microsoft factory then this whole certificate infrastructure as it's been proposed would give you the software version or the cryptographic version of shrink-wrap and maybe that's what you really want, but that's not the dream of Internet commerce. The dream of Internet commerce is you don't need the huge physical infrastructure and capitalisation barriers that you need in the real world. You can have tons and tons of small merchants.

So what we think Alice was really trying to figure out, or what she should be trying to figure out when she goes through deciding "that this was signed by the right key", that this software she downloaded and is considering paying for or considering using in her machine, it might be her machine, what she's really trying to figure out is does this key, or really does the signing key that corresponds to this verification key comply with her policy. And here's an example of what her policy might be. So in other words whoever Bob is, if he's somebody she should trust, whatever she means by trust.

So here's what she might mean by trust. She might want to know that any merchant that she purchases something from over the Internet is careful about his customers' privacy, so the EFF is the electronic frontier of verification, they are primarily extreme privacy advocates, maybe I'm being over-complimentary, I don't know. So a wary customer might want to get a certificate, or actually the term I prefer is credential, that whoever signed this thing was approved by this agency, namely the EFF for privacy handling, it will not go and sell its customer data to somebody nefarious organisation.

She might care about safety, she might not want to download some software from some merchant she doesn't want to, install it and run it and not care whether it's going to eat her machine, or at least that's the theory. As far as I can tell people do that everyday, all the time, but we security researchers are really intent on providing mechanisms to make sure people don't have to do that. OK, but that would be a good policy. So I couldn't think of an agency that she would go to for credentialling for safety, because I think actually almost nobody knows anything about how to look at a piece of code and decide whether it's going to eat your machine or not. So SMB stands for Stephen Michael Bellovin, who's an AT&T person who I know very well and whom I would definitely trust for such purposes, and I think that he would look at this code and he would tell you don't run it, just say no, that would be his credentialling.

She might also care that these credentials were issued fairly recently. So this would be a not very complex policy, this policy is actually addressing the kind of concerns that people do care about, about privacy, about safety or some kind of domain expertise, about recentness, timeliness of information. Notice that these are realistic policy elements and none of them has anything to do with names. What she doesn't need to know is the name of the guy whose public key is this sequence of digits and if she did know it, it wouldn't answer her real question.

So that's our motivating example, as I say I hope that's a straw man but I don't know, maybe there are people who really believe in X.509 certification stuff.

Alright so here's what we have in mind that what we really want in a situation like purchasing software on the Internet or more general e-commerce situations, and many other situations as well, is something called a Trust Management System. You want to be able to feed in a request like download this software and install it and run it and pay for it, or maybe that would be a sequence of requests, a policy saying what you require of something that you, before this request should be satisfied and a pile of credentials.

OK, and then the Trust Management System is supposed to decide does this request supported by these credentials satisfy the local policy. Another way of saying that, do these credentials prove that this request complies with my policy. So I put AKA Rights Management, actually I would say Rights Management is as it's understood as a term of art, you know Rights Management System, Licence Management Systems would be an example of that. So that was what we set out to do and two years ago Matt Blaze, Jack Lacey and I published our first paper on this stuff in the Oakland Conference in 1996 and here's what I think in retrospect. I don't know actually whether all of these points are raised in the original paper, but I think most of them are at implicitly and I think perhaps explicitly. This is what I think are some of the crucial questions that you would have to answer if you wanted to design a Trust Management System that does what that previous transparency says one should do.

So the first question that the straw man example is set up to point us to is, are you really doing authentication or are you doing authorisation? Are you interested in who signed this request or are you interested in whoever signed that request is he authorised to do the thing that he signed. Now Roger, who unfortunately is not here, I was just about to get some reasons, controversy, Roger said in his opening remarks is that what he thinks we should be aiming at is figuring out how to track down the bad guy afterwards and make sure that he is held responsible for whatever security breach he caused, and I guess I disagree with that. Not that that is a bad thing, I'm saying that really what everybody wants to do in a security system and in that in fact I think it's unrealistic to expect that you will be able to do that.

So these public keys certificates that bind names to keys and that have gotten so much attention over the years, why do they get so much attention when it seems pretty obvious whenever I put up that straw man example I get lots of nots, it seems pretty obvious to people that even if you could sort of resolve all the name key bindings you wanted it wouldn't really answer your security question. It wouldn't answer your question about compliance with policy. Why is it that people nonetheless put this immense amount of effort into certificates that allow you to do name key bindings. Because I think a lot of people really maybe without thinking it through, maybe with thinking it through, are trying to do what Roger suggested, they're not thinking about authorising an action before the fact, they're thinking about accountability for an action after the fact.

Stewart Lee: I think you misread Roger.

Reply: Yes, what did Roger mean? He's not here, we can decide what he meant.

Stewart Lee: I've talked many times on this subject and what he wants to do is increase the street price of your medical records to a point where it's not worth somebody's effort to attempt to get them.

Reply: Because he assumes that he'd get caught. That's what you're doing.

Carl Ellison: That's just way of increasing the price. What I think was important about what Roger said was that you want to deal with the cost of the security violation and increase the cost of the security violation without necessarily prohibiting it entirely.

Reply: Right OK so I'm going to agree with that.

Stewart Lee: Absolute prohibition is absolutely impossible.

Reply: OK so I would agree with that and that actually is a sort of a vital point to the one which is that I think accountability after the fact is fine but it's not really what you want to do in a lot of e-commerce systems. You really do not want, you the user of such a mechanism really does not want to be back to the fact and track down whoever signed something that caused a wrong action.

Carl Ellison: Perfect example of that is what happens in grocery stores where they used to accept cheques if you would write your drivers licence number on it, but now they accept ATM cards.

Reply: Right, perfect, exactly. Who the hell would really want to go and track down via the DMV somebody did something wrong on his grocery bill, you want to have it authorised beforehand.

Stewart Lee: Furthermore I had a student who parents were blind and they kept going to the store and writing cheques and being asked for their driving licence!

Bill Harbison: Let me put a counter point, if we had a perfect audit trail, perfect in the sense that whatever happened was recorded and was not tampered with, even if you did the wrong thing it didn't matter, you had it there in hardware, and it was known that this existed, would the existence of such a system prevent people doing things just because they knew that if anything was questioned they would come back and say we know it can be proved that this transaction occurred.

Reply: Perhaps. I still think that that's not what you really want. Let me also make the point that we intend there to be a discussion, we not each going to talk for half an hour, we intend there to be a discussion and we can have this discussion then. Let me just directly answer your question. I think that perhaps that would, that certainly would deter some things but I think first of all the person running this system still may not want to do it that way because he may not want to approach his security problem by saying I'll keep track of everything and then I'll go and pay the full cost of going back and auditing everything and tracking down bed bugs. He may much prefer the authorisation before the fact because it's easier for him. The other thing is I don't think that we really want to require everybody to implicitly consent to having everything that he does and every transaction he requests, recorded. I think you might want to allow anonymous credentials that still have cryptographic strength of a very precise sort, but this is not a pro anonymity talk, that's a different talk.

Bill Harbison: I agree, my comment was in the context of what I think was at the heart of what Roger was saying as opposed to ...

Reply: OK, boy I'm really sorry that I did that, everybody feels they have to defend Rog. OK.

So what you really want to do is decide are you really interested, do you want certificates that will allow you to figure out who signed what or do you want credentials that will allow you to decide is this request OK according to my policy.

You wanted another interesting general question in this Trust Management System that I had on my previous slide, where you feed requests and credentials and policy in and you get a comply/doesn't comply answer out. Should there really be an application independent general purpose box that does what that Trust Management System says it does. We think so, we think it's very much worth the effort of trying to figure out what does proof of compliance mean and build it into such a general purpose Trust Management System and I said something about that in my position paper in these proceedings. I also have in this Financial Crypto paper a longer discussion of why we think that's a good idea, but that's a very controversial idea. A lot of people tell me, that just doesn't make sense, there is no application independent notion of proof of compliance, just like you're saying Joan there shouldn't be a global notion of the certificate, you should think about local policy. Well the same thing applies to proof. I guess I don't really agree with that, but I don't feel quite as adamant about it as I do on the 'what you really want is authorisation' notion. There are system architecture questions, namely, I said you want to feed this stuff into Trust Management engine, well exactly where do you divide responsibility and have the separation of duty between the call-in application and the Trust Management System.

I've already touched on how would you define the proof of compliance given that you want a general notion of proof of compliance. Should your credentials and policies be programmable, should they be fully available or partly available. Notice that a traditional name key binding certificate is not available at all. It's just a piece of paper. What language would you want to programme these things in? What are performance considerations? There's a whole lot of general research questions, and that I think is basically the contribution of that paper.

So, the decisions that we made in that paper, were that we were going to have direct authorisation of keys, so a credential could say I is credential in agency confer the authority on this key to sign requests of the following four, not I certify that this is box key. We have a general purpose Trust Management System in particular we've put a lot of effort into the policymaker Trust Management System. For system architecture questions we emphasise two things. One was minimalism. We wanted the Trust Management System to do as little as possible and do it very well and to analyse it very completely, and in particular we chose not to have the Trust Management System do credential fact checking and not to have the Trust Management System do signature verification, do the arithmetic part of the signature. What I mean by local is basically you can't get policymaker to do anything unless you feed it at least one local policy statement. There is no assumption that there is some globally trusted authority of any kind. Doesn't mean that you can't use an X.509 certification authority, or certification hierarchy. If you want to you can but you have to explicitly say that you want to. And we have programmable credentials and policies and I won't go into this here, I can spell it out off line for people who are interested. Basically credentials and policies are essentially the same type of object and this facilitates a very interesting and useful type of delegation. If I want to say my policy is Carl's policy, I trust Carl, he knows everything about security, my policy is Carl's policy, then I can say that in a policymaker assertion. Carl can take his policy and sign it and then his policymaker policy becomes the policymaker of credential and it all can get fed into the same proof system and everything works just fine.

So that's what I wanted to say about ...

Virgil Gligor: Do you have a don't care statement?

Reply: No, and that is a deep question, speaks to the monotonicity issue, discussed in detail in this paper. All you can do in policymaker, and I believe in anything that's going to be discussed in this session, is approve things. You can approve things or you can fail to approve things. I am about to mention one thing in which you can approve, disapprove and abstain but once you introduce that disapprove business or anything other than simply approve or do nothing you get a much more complicated proof of compliance notion. And I can actually prove it. I can prove that the complexity of proof of compliance is higher. I can prove that it's NP hard if you don't have monotonicity and that it's polynomial if you do. That's all formalised in that paper.

So that's the basic idea of, that goes with these sort of the design decisions that we set out to test. Now, we have implemented it and I believe that the most up-to-date implementation policymaker is by Angus so he can tell you all the details of what's available and implemented. We've used it in several applications, including the commercial, like an AT&T offering, in music distribution to do rights management which is apparently an immensely complicated thing in the music business. I didn't know that but Jack Lacey, Dave Mair and Jim Schneider used policymaker to do licence management on a musical offering and apparently you really need something very subtle to do licence management in music, there's lots and lots of rights holders and if anybody wants to play any piece of music even for personal use, there are right questions, but certainly for commercial use. I find this amazing.

So, I just want to say what I think an interesting question was that came up in this Internet content labelling application which was the only one I was really heavily involved in. I did it with Matt Blaze and Paul Resnick, one of the inventors of pix content labelling system and ... And what came up is a fact about how you would do these labels, how you would do trust management is a content labelling system. It says that this architectural boundary question is actually very subtle. Suppose you had this e- commerce system, and the idea is that the user and merchants and the credential bureaux and everybody somehow comes up with a request, a pile of credential and a policy, feeds it to the Trust Management System and gets back a yes/no decision. So this trust management is done here and everything else is done there. That's what we were aiming at. So consider the following example. This user is interacting with this Internet shopping mall. Now you might think no-one would buy a car using an Internet shopping mall, but I was actually at a talk at Rutgers when someone told me that he did that. Amazing!

So suppose someone really wanted to do this. He wanted to issue the following request to his browser. He said buy a Ford Taurus from this guy, assuming this guy tells you that he's going to charge you less than $20,000 or at most $20,000, I don't know if that's realistic, I don't actually own a car. So why would you feel confident doing that when you've got this nifty Trust Management System and you've got this great policy. In addition to privacy you required that some good credentialling agency, say Consumer Reports, tells you that if this merchant says it costs only $20,000 then it costs only $20,000, he's honest, he doesn't place dishonest labels on his product. Well, I think this is the problem, OK, it's not clear how you would actually do this with a Trust Management System. Why, because according to your policy, namely your privacy policy, you're not even supposed to go to this merchant and get this label, this price label, until you've already checked that this merchant satisfies your privacy policy. So you really have to do two trust management calls, you have to do two calls to this guy. If the first go get the credentials that prove this merchant's OK, then go to the merchant and do another check with the Trust Management System and the reason I think this points out how difficult all that is, is that dividing it into those two things is itself a trust decision. OK, knowing that you should first go to this guy and then do some computation about whether his credentials prove that that guy is OK, and then go to that guy and do so computation about whether you should buy something from him, I think that is itself a trust decision. So these architectural boundaries are not something we've completely solved.

Let me just wrap up by saying there's a lot of other stuff that we haven't completely solved either and is this really, overall, is this really a good idea, is this kind of thing ultimately going to be widely used, I think it's still an open question. I think it's promising but we certainly haven't proved it yet. We have our proof system which we designed from scratch, and we can prove a lot, we can make formal statements about, but saying this is the right definition of proof of compliance, that's a subjective statement, I'll never be able to convince you of that if you like some other proof system better. I have a student experimenting with a more conventional logic base proof system and this is an ad hoc proof system. The non monitoricity question has already been alluded to, what language should you use to write these credentials, that's a wide open question, and I think basically the name question which is sort of the same as this question is 'What's the right test bet for this?' What would really show convincingly that having a general notion of proof of compliance and a general Trust Management System is a good idea.

KeyNote: Trust Management for Public-Key Infrastructures
(Position Paper)

Matt Blaze[1], Joan Feigenbaum[1], and Angelos D. Keromytis[2]

[1] AT&T Labs – Research
180 Park Avenue
Florham Park, NJ 07932 USA
{mab,jf}@research.att.com

[2] Distributed Systems Lab
CIS Department, University of Pennsylvania
200 S. 33rd Str., Philadelphia, PA 19104 USA
angelos@dsl.cis.upenn.edu

Abstract. This paper discusses the rationale for designing a simple trust-management system for public-key infrastructures, called *KeyNote*. The motivating principles are expressibility, simplicity, and extensibility. We believe that none of the existing public-key infrastructure proposals provide as good a combination of these three factors.

1 Introduction

Trust management, introduced in the PolicyMaker system [2], is a unified approach to specifying and interpreting security policies, credentials, and relationships that allows direct authorization of security-critical actions. Security credentials (which subsume the role of "certificates") describe a specific delegation of trust among public keys; unlike traditional certificates, which bind keys to names, trust-management credentials bind keys to the authorization to perform specific tasks. KeyNote provides a simple notation for specifying both local security policies and security credentials that can be sent over an untrusted network. Policies and credentials, called "assertions," contain predicates that describe the trusted actions permitted by the holders of specific public keys. A signed assertion that can be sent over an untrusted network is called a Credential Assertion. Credential assertions, which subsume the role of certificates, have the same syntax as policy assertions with the additional feature that they are signed by the entity delegating the trust. A KeyNote evaluator accepts as input a set of local policy assertions, a collection of credential assertions, and a collection of attributes, called an "action environment," that describes a proposed trusted action associated with a set of public keys. KeyNote determines whether proposed actions are consistent with local policy by applying the assertion predicates to the action environment.

Although the basic design of KeyNote [1] is similar in spirit to that of PolicyMaker, KeyNote's features have been simplified to more directly support public-key infrastructure-like applications. The central differences between PolicyMaker and KeyNote are:

- KeyNote predicates are written in a simple notation based on C-like expressions and regular expressions.
- KeyNote assertions always return a boolean (authorized or not) answer.
- Credential signature verification is built in to the KeyNote system.
- Assertion syntax is based on a human-readable "RFC-822"-style syntax.
- Trusted actions are described by simple attribute/value pairs.

SPKI/SDSI [4] also essentially attempts to do trust-management for public-key infrastructures. We believe that KeyNote provides a better solution, because it was designed to be simpler, more extensible, and more expressive than SPKI (and certainly than X.509 [7]). In the following sections, we expand on these three design principles.

2 Simplicity

Simplicity in the design of KeyNote manifests itself in various aspects of the system:

- Narrow focus.
  KeyNote aims to provide a common, application-independent mechanism for use with application-specific credentials and policies. Each application (or class of applications) will develop its own set of attributes, with application-specific credentials and policies created to operate on them. Other approaches include name-based schemes (such as X.509 [7]), in which the infrastructure aims to provide a common application-independent certificate with each application left to develop its own mechanism to interpret the security semantics of the name, and SPKI/SDSI [4], which attempts to provide both an authorization and a naming mechanism. Both systems are unnecessarily large and complex (because secure naming and authorization are orthogonal).
- Easily describable system.
  The basic KeyNote document [1] is 15 pages long, including an extensive examples section. We believe that such a compact document makes the system easy to understand. Contrast this size with other systems' documentation (especially those produced by committees!)
- Easy to understand and create assertions.
  The KeyNote assertion format is human-readable, ASCII-based. Furthermore, the notation used to describe the trusted actions is based on C-like expressions (arithmetic, string, and boolean operations), which is widely known and used. Although we intend to provide an easy-to-use graphical user interface, it is possible to write a KeyNote assertion using just a text editor (except for the signature of course).

– Easy to implement.
  Our reference implementation is less than 1500 lines of *C* and lex/yacc specification and was written with readalibity in mind. The small code size argues for robust and relatively bug-free implementations. The same cannot be said of all other schemes.

3 Expressibility

The KeyNote language is designed to make it easy to express and evaluate the kinds of policies and trust delegations that occur in "public-key infrastructure" applications.

KeyNote syntax is minimal and reflects the system's focus on delegation of authority. The basic element of KeyNote programming is the *assertion*. Assertions are the mechanism by which a key (or collection of keys) is authorized to perform various trusted actions. Assertions are used to specify local policy. The same assertion syntax is also used to provide signed credentials in which one party defers authorization to another. Thus security policies and credentials share a common syntax, and policy can be used as a credential simply by signing it. Assertions are written independently from one another and are essentially autonomous programs that do not communicate directly with one another or depend on other assertions or externally-defined data structures.

Assertions are designed to be easy for humans and computers to write and understand. In particular, simple authorizations have simple syntax, and it is usually easy to determine what an assertion does simply by reading it. The function of an assertion is to allow an entity to authorize another entity to perform specific trusted actions. An assertion is divided into sections that reflect the logical components of such an authorization. One section identifies the authorizer (either local policy or, in the case of credentials, the key that signed it). Another section, the "key predicate," describes the key or collection of keys being authorized. Finally, the "action predicate" describes the action being authorized.

Assertions are designed to require only minimal computation to evaluate. Key predicates and authorization predicates are based on simple C-like expressions that match against action environment attributes. In particular, there are no loops or function calls, and an assertion can be parsed in a single pass. The expression semantics are rich enough to allow complex regular expression matching and numeric evaluation but constrained enough to allow a KeyNote evaluator to be built into embedded applications or operation system kernels.

Our design philosophy for KeyNote thus departs from that of PolicyMaker, in which assertions can be arbitrary programs. PolicyMaker is designed to provide a general trust-management framework across a wide range of applications, at some expense of efficiency. KeyNote, on the other hand, provides somewhat simpler syntax and semantics, aimed specifically for building public-key infrastructure applications, at some expense of generality.

4 Extensibility

Two important components of a trust-management system are the assertion syntax and the compliance-checking algorithm. (Recall that "compliance checking" is the process of deciding whether a set of credential assertions prove that a request complies with a policy assertion.) In the PolicyMaker trust-management system [2,3], both the assertion syntax and the compliance-checking algorithm are proper generalizations of the corresponding components of KeyNote. This implies that KeyNote is highly extensible; indeed, its natural extension has already been implemented. An application that needs more general assertions than KeyNote provides can easily upgrade its trust-management engine: It can switch from using the KeyNote compliance checker to using the PolicyMaker compliance checker, write its new, more general assertions in PolicyMaker, *and continue to use its old KeyNote assertions, because they are compatible with PolicyMaker assertions and can be processed by the PolicyMaker compliance checker.*

The PolicyMaker notion of "proof of compliance" is beyond the scope of this paper; it is discussed in full detail in [3]. Here we briefly explain one way in which PolicyMaker assertions generalize KeyNote assertions and why this generalization might be useful. A PolicyMaker assertion is a pair (f_i, s_i). The "source" s_i is basically identical to the KeyNote SIGNER field. The function f_i is a general program that may be written in any language that can be "safely" interpreted within the trust-management environment; in particular, f_i may be a KeyNote assertion. Rather than simply returning TRUE or FALSE, PolicyMaker assertions produce sets of "acceptance records." A record has the form (i, s_i, R_{ij}) and means that "assertion number i, issued by source s_i, approves action string R_{ij}." The action string *may* be the request that was fed to the trust-management system by the calling application, or it may be some related action-string that makes sense in the context of this attempted proof of compliance. Each time it is executed during an attempted proof, an assertion receives as input all of the acceptance records that have been approved so far by the policy assertion (f_0, POLICY) or by any of the credential assertions (f_1, s_1), ..., (f_{n-1}, s_{n-1}). Such non-boolean assertions are useful in several standard trust-management constructions, including those that require explicit control over delegation depth. See [3, Section 3] for more details.

Ellison *et al.* note that some applications may require more expressive power than the SPKI/SDSI certification framework provides, and they suggest the use of PolicyMaker to achieve this additional power. More precisely, [5, Section 7.3] contains the following suggestion: "For any trust policy which the full SPKI 5-tuple reduction can not express, one must write a policy interpretation program and PolicyMaker provides a language and body of examples for that purpose. The result of the PolicyMaker execution can be a 5-tuple to be used within an SPKI 5-tuple reduction." The observation that a PolicyMaker assertion may produce an acceptance record whose action string R_{ij} encodes a SPKI 5-tuple is a good one, and this may indeed be a reasonable way to extend SPKI. However, the precise relationship between the PolicyMaker definition of "proof of compliance" and the definition given by SPKI 5-tuple reduction has not yet been formally analyzed. (This is a potentially interesting direction for further research.) Thus we cannot formally characterize the additional power that this use of PolicyMaker would bring to the SPKI/SDSI framework or assess the ease with which such an extension could be implemented.

5 Conclusions

We have presented the design principles for KeyNote and contrasted our design with other existing proposals. We believe that the combination of simplicity, expressibility, and extensibility makes KeyNote well-suited for trust-management in public-key infrastructure.

For more information about KeyNote, please read the Internet Draft [1].

References

1. M. Blaze, J. Feigenbaum, and A. D. Keromytis, "The KeyNote Trust Management System," work in progress. Internet Draft, April 1998, `http://www.cis.upenn.edu/~angelos/draft-angelos-spki-keynote.txt.gz`.
2. M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," in *Proceedings of the 17th Symposium on Security and Privacy*, IEEE Computer Society Press, Los Alamitos, 1996, pp. 164–173.
3. M. Blaze, J. Feigenbaum, and M. Strauss, "Compliance Checking in the PolicyMaker Trust Management System," in *Proceedings of the 2nd Financial Crypto Conference*, Lecture Notes in Computer Science, Springer, Berlin, 1998, to appear. Available in preprint form as AT&T Technical Report 98.3.2, `http://www.research.att.com/library/trs/TRs/98/98.3/98.3.2.body.ps`.
4. C. Ellison, *A Simple Public-Key Infrastructure*, `http://www.clark.net/pub/cme/html/spki.html`.
5. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, "SPKI Certificate Theory," `http://www.clark.net/pub/cme/theory.txt`
6. R. Rivest and B. Lampson, *SDSI: A Simple Distributed Security Infrastructure*, `http://theory.lcs.mit.edu/~rivest/sdsi11.html`.
7. Consultation Committee, *X.509: The Directory Authentication Framework*, International Telephone and Telegraph, International Telecommunications Union, Geneva, 1989.

Discussion Session – Trust Management

Virgil Gligor: I also think that the talks point out something which was pointed out by I guess Jim Morrison about 1971, which is the cyclic dependency between security and access control. And I think really that's one of the few very fundamental observations about security in general. I mean, at some point to do authentication you need access control, and most people recognise that to do access control you need authentication.

Bill Harbison: There's also the view of the protocol, which Roger didn't say this morning, which is that it's a process of relating parts of the system that are not otherwise directly related.

Michael Roe: We are actually all agreed on which security protocol we are talking about here: it's a bundle of certificates followed by some signed data. The big argument is about what on earth it is you did and what the semantics are.

Stewart Lee: And what it all means.

Michael Roe: And you could read my doctoral dissertation which says exactly the same thing.

Stewart Lee: Well, had I not examined your doctoral dissertation I might not have asked that question.

Virgil Gligor: The thing that Carl's talk points out, at least to me, a question that I cannot resolve right now, which is, and I'm listing it as a conjecture, is it possible to do security, in a generic sense, in an open system without any notion of global trust or global assumption or global anything.

Joan Feigenbaum: I might say, I think I'm a little confused. But I think I know what you're asking, which is suppose you don't have any kind of global agreement on the meanings of these authorisations. Then as far as I'm concerned what that will mean is that any requester who comes to me, who controls the policy in my environment and says I request to do something and here are my credentials, won't be able to do something that I don't have a policy about, I'll just say no.

Virgil Gligor: Security means denial of service.

Joan: No, that's not denial of service!

Virgil Gligor: In my opinion, and I may be all wrong about this, in an open system you have to have at least some form of global agreement, possibly on registration of names, perhaps of one name, but you have to have, in my opinion, something that's global.

Angelos Keromytis: It's in my best interest to agree, I mean if I'm an attacker of an open system it's in my best interests to agree on the communication protocol principles. I want to use Telnet to connect to whoever, but it doesn't mean that it's in my best interest to follow your trust certification to the letter if I can abuse it a bit.

Carl Ellison: I think I have a new understanding of what you're asking. Let me give it. In all of this work, well at least in KeyNote and SPKI, we do rely on something global, on a global name system. The global name system we rely on is called the public key. We are making an observation that this randomly generated public key is globally unique.

Virgil Gligor: It's generated by unique authority?

Carl Ellison: No. Randomly generated by the keyholders. And by a random process. And that is the only globally unique thing that we have. But that's enough.

Bruce Christianson: If it's long enough.

Virgil Gligor: I think that is fair enough, but there is global mathematics here. There is something that's interpreted by systems in the same way.

Bill Harbison: It is also the case that an intersection of a number of closed systems, no matter how large, is not necessarily open, it may very well be closed.

Joan Feigenbaum: But that makes it good.

Bill Harbison: Indeed so, that's the point, that although we talk about open systems many of the cases at which we're actually looking are interconnected closed systems, they are not in fact open.

Virgil Gligor: I believe that, but I think there is a lot of value in what you are doing. In fact I would recast the whole thing in reducing trust to perhaps this globality of the public key. The point is that there is value in that process, there is security value in it, there is trust no matter how you call it. Whatever you call trust, minimizing it makes sense. Why does it make sense? You minimize what I call trust-liability. Trust means liability. You rely on something, you depend on somebody else. So minimizing that is a positive thing because you minimize liabilities. But I wouldn't argue that there's absolutely nothing that's global, no assumptions. I would argue, perhaps I'm repeating myself, our minimizing it is something that's valuable.

Rapahel Yahalom: Minimizing liability may not be a goal which everybody will strive for. Some corporations like banks live on accepting liability. But that's just an aside meant as a supercilious comment. I do have a question to all the three previous lecturers, particularly to Joan, since she made the comment, but Joan, wear your hat as a researcher as opposed to a member of a commercial organisation like AT&T.

Joan Feigenbaum: That's a great distinction.

Rapahel Yahalom: If you are able to make that distinction. But seriously you made the point at the end saying, we still haven't proven this is a good idea. I think "prove" is a quote-unquote, however my question as a researcher is, what directions do you have in mind? How can one prove something like that about a trust management system. One can ask that about any application system, but are we going to let the market forces make a decision? In that case we might one day conclude that Internet Explorer is better than Netscape Navigator. But are there any more scientific approaches?

Joan Feigenbaum: I don't know whether the other two speakers even agree this is still an unproven idea, maybe they think it's been proven to their satisfaction that general purpose trust management is a great thing. To answer your very specific question, my future directions if I keep working on this will be to try to improve and consider alternatives to the proof of compliance mechanism which is specified and analysed in that paper. And I should make clear that that's the aspect of this system that I worked on most. And the other will be, and this is what I think is actually much more important, is proving that this is a good idea. Which would have to mean that multiple different applications that process signed requests and do proof-of-compliance use it, and that somehow the developers, the administrators and the users feel they've gotten some value added from the trust management system as opposed to just doing some roll your own version of proof-of-compliance. So experimental research which, you know, there might be a market component to that, maybe if the experimentalists in this audience could tell me, without having a market that either accepts it or doesn't accept it, how would you do an experiment that would demonstrate that this is a good idea.

Bill Harbison: But your compliance mechanism is effectively, by being divorced from an application, syntax directed rather than semantics directed and therefore loses, in a commercial domain, a lot of it's value, does it not?

Joan Feigenbaum: Not clear, it might gain a lot.

Bruce Christianson: That's actually one of it's great strengths, because all sorts of people who are unable to agree about the correct semantic context in which to conduct a security policy will nevertheless say that this tool manipulates bit patterns in exactly the right way.

Joan Feigenbaum: We tell you how we take your assertions, that is credentials and policies that you understand, and sew them together. We tell you exactly how we do that.

Bill Harbison: This brings to mind people who, not wishing to use the word trust, defined a new word called jurisdiction.

Application-Oriented Security Policies and Their Composition
(Position Paper)

Virgil D. Gligor[1] and Serban I. Gavrila[2]

[1] Dept. of Electrical Engineering, University of Maryland, College Park, MD 20782
[2] VDG Inc., 6009 Brookside Drive, Chevy Chase, MD 20815

Abstract. We define the notion of the application-oriented security policy and suggest that it differs from that of a system-level, global security policy. We view a policy as a conjunction of security properties and argue that these properties are not always independent and, hence, cannot be analyzed (e.g., composed) individually. We also argue that some necessary policy properties fall outside of the Alpern-Schneider safety/liveness domain and, hence, are not subject to the Abadi-Lamport composition principle. We suggest several areas of research in policy definition, composition, and administration.

1 Application-Oriented Security Policies

We distinguish an application-oriented (a-o) security policy from a system-level, global security policy by three characteristics, namely, (1) locality of enforcement, (2) mandated composability with other a-o policies, and (3) mandated compatibility with the application itself.

Locality of Enforcement. Enforcement of a policy can be local or global. Enforcement of a policy in an application is *local* if the policy can be completely and correctly enforced within the application. Enforcement of a policy is *global* if the policy must be uniformly enforced for all applications under its control. Although in systems supporting multiple applications, an a-o policy must also be composable with an underlying (more) global policy in the sense that the latter must enable the local enforcement of the former, the a-o policy need not also be enforced within the global policy. For instance, a database security policy must use its underlying operating system's policy to protect its objects from unauthorized access by other applications, but the operating system policy does not have the obligation to also enforce the database policy. In contrast to the local enforcement of an a-o policy, a global policy must be enforced *uniformly* for all applications under its control or, else, it may be ineffective for any applications.

Mandated Composability. Mandated Composability means that an a-o policy must be composable with any and all other a-o policies with which it coexists within a system. This is both a separate concern and obligation. It is a separate concern because global policy support of individual a-o policy enforcement (i.e., "hierarchical" [8,10] a-o policy composability with the underlying global policy) cannot always guarantee composability among a-o policies, since the global policy is oblivious to interactions among applications. It is a separate obligation because interactions among a-o policies may produce distinct, emerging a-o policies. As a consequence, ("lateral" or "external" [6,7]) composability must be part of a-o policy definition and administration.

Mandated Compatibility. Mandated compatibility of an a-o policy with the application itself means that the a-o policy may not unjustifiably deny the execution of an application for which the policy was developed. For example, the access-management policy may grant users too few, or inconsistent, permissions for an application. Specifically, access management may grant a user the permission to invoke an application operation and insufficient permissions for operation execution, or vice-versa, thereby unjustifiably denying application execution. Should this happen, either the application or the policy definition, or both, must be changed to achieve compatibility (viz., Figure 1). Otherwise, it would be both senseless and ironic to create a specially tailored policy for an application only to discover that the application could never work under the constraints of its policy.

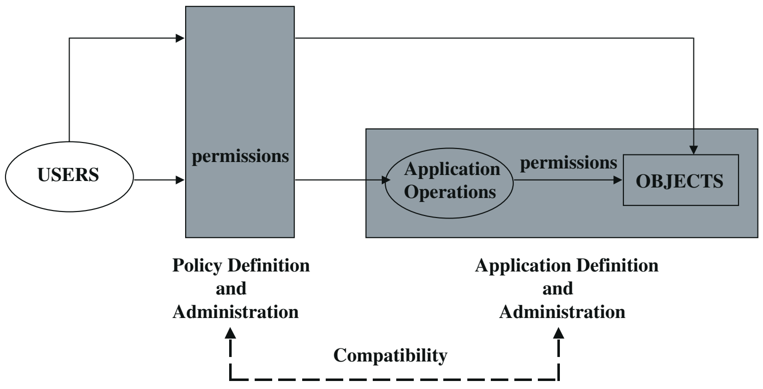
USERS
permissions
Application Operations
permissions
OBJECTS
Policy Definition and Administration
Application Definition and Administration
Compatibility

Fig. 1. Mandated Compatibility

2 Structure of Application-Oriented Policies

We view security policies as conjunctions of security properties. However, these properties are *not* independent [3,4,11]. To illustrate the notion of dependency among policy properties, consider the following three types of properties: *access-attribute* (AT), *access-authorization* (AA), and *access-management* (AM) properties. Access attributes include subject and object attributes (e.g., user, group, role, location identifiers, secrecy and integrity levels, time-of-access intervals), and AT properties typically establish invariant relationships among these attributes (e.g., lattices of secrecy and integrity levels, user-group membership invariants, inheritance of role permissions). AA properties determine whether a subject's current access attributes satisfy the conditions for accessing an object given the current object attributes. In contrast, AM properties include conditions under which subjects are granted or revoked attributes for accessing objects, subjects and objects are created or destroyed, objects are encapsulated within protected subsystems. In general, AM properties bring the system to desired states, whereas AA properties ensure that the system remains in desired states.

A typical dependency arises between AA and AM properties because the values of the attributes used by the former for access decisions are determined by the latter (i.e., AA properties have a "uses" dependency [4,11] on AM properties). Hence, whether an AA property holds in a state depends not only on the property (predicate) itself, but also on the defined AM properties. Both AA and AM properties have "uses" dependencies on AT properties. Furthermore, AM properties also have "uses" dependencies on AA properties whenever object attributes themselves are treated as objects and, hence, become the target of access authorization. Techniques to redefine policy properties to avoid this cyclic dependency, and other undesirable ones, are presented elsewhere [4].

Groups of policy properties may also depend upon individual properties used in the same conjunction. For example, the property $P = AT \wedge AA \wedge AM$ depends on an AM property, denoted by $Admin(P)$, which requires that administrative commands have the ability to bring the system from an arbitrary state (or, alternatively, from any state) to a state that satisfies P. Unless $Admin(P)$ is satisfied, the policy defined by property P is empty because the system operating under P can neither start from, or recover to, a secure state [9], nor reach a secure state from another secure state under administrative control.

For *application-oriented* policies, the composition of independent security properties does not necessarily define a *usable* policy. To be useful, property $P = AT \wedge AA \wedge AM$ must be compatible with the application for which it was designed.

3 Composition of Application-Oriented Policies

We define a system by a state machine model. We denote the set of states by $STATES$, the set of subjects by $SUBJECTS$, the set of users by $USERS$, the set of operations by $OPERATIONS$, and the set of objects by $OBJECTS$. Each *state transition* is defined by a command of the form $op(s_1, S, obj, s_2)$, where the subject S performs the operation op on one or more objects obj, thereby changing the current state s_1 to s_2.

A *command sequence* is $op_1(s_0, S_1, obj, s_1) \cdot op_2(s_1, S_2, obj_2, s_2) \cdot \ldots$, where "$\cdot$" is the concatenation operator, and s_0 is the *start state*. We denote the set of start states by $STATES_0$. A finite command sequence σ may be extended to an infinite one by adding "no-op" commands, that do nothing and preserve the current state. The extended command sequence is denoted by $\hat{\sigma}$.

We denote the set of all command sequences with the start state in $STATES_0$ by Ω_0. A *tranquil command* is a command that does not alter the security attributes of the subjects and objects (e.g., creation, deletion, or update of permissions, users, roles). A *tranquil command sequence* is a command sequence consisting only of tranquil commands. We denote the set of tranquil command sequences with the start states in $STATES_0$ by Σ_0. Whenever $STATES_0 = STATES$, we drop the subscript $_0$.

A *system* is a set of command sequences with start states in $STATES_0$. A *secure state* is a state that satisfies some "state properties". A *secure command* is a command that satisfies some "transition properties". A *reachable state* is a state appearing in a command sequence of Ω_0.

An *application* is a tuple $App = [ObjSet, OpSet, Plan]$, where $ObjSet \subseteq OBJECTS$ and $OpSet \subseteq OPERATIONS$. *Plan* is the *execution plan* of the application and consists of a finite set of pairs $\{(obj_i, op_i)|i \in \{1, \cdots, n\}\}$, where n is a natural number, obj_i is one or more objects of $ObjSet$, and op_i is in $OpSet$. Other types of execution plans may include order or exclude redundant operations and privileges to objects to help satisfy the "least privilege" principle. For simplicity, we omit these types of plans.

Given two applications $App_i = [ObjSet_i, OpSet_i, Plan_i]$, $i = 1, 2$, we denote the new application $[ObjSet_1 \cup ObjSet_2, OpSet_1 \cup OpSet_2, Plan_1 \cup Plan_2]$ by $App_1 \sqcup App_2$.

A command sequence $\sigma \in \Sigma_0$ *executes* the application *App* (or *App* is executed by σ, or σ is an execution of *App*), if for any pair (obj, op) in *App*'s execution plan there is a command $op(s_k, S, obj, s_{k+1})$ in σ. We also use *App* to denote the set of all executions of *App*.

For an a-o security policy

$$\mathcal{P} = P \wedge Admin(P) \wedge Compat(P, App),$$

where $P = AT \wedge AA \wedge AM$, we define $Admin(P)$ and $Compat(P, App)$ as follows.

Definition 1. *$Admin(P)$ is satisfied if and only if $\forall s \in STATES, \exists s_0 \in STATES_0, \exists \omega \in \Omega : \omega$ starts in $s \wedge \omega$ reaches $s_0 \wedge \hat{s_0} \in P$; i.e., starting in an arbitrary state, the administrative commands have the ability to bring the system to a state that satisfies property P. (Alternatively, we can define $Admin(P)$ as follows: $\forall s \in STATES, \forall s_0 \in STATES_0 : s_0 \in P, \exists \omega \in \Omega : \omega$ starts in $s \wedge \omega$ reaches s_0).*

Note that the predicate "ω reaches s_0" of $Admin(P)$ is not trivially satisfied; e.g., the system may not provide all the administrative commands to ensure that certain states of $STATES_0$ can be reached.

Definition 2. *Compatibility of P with App is any one of the following six alternatives:*
1. P is compatible *with App if there is a $s_0 \in STATES_0$ and a $\sigma \in P$ starting*

in s_0 *that executes App. We denote this property by* $Compat(P, App)$.
2. P *is* strongly compatible *with App if for each* $s_0 \in STATES_0$ *with* $\hat{s_0} \in P$ *there is a* $\sigma \in P$ *starting in* s_0 *that executes App. We denote this property by* $Compat_{\mathrm{S}}(P, App)$.
3. P *is* totally compatible *with App if for each* $s_0 \in STATES_0$ *there is a* $\sigma \in P$ *starting in* s_0 *that executes App. We denote this property by* $Compat_{\mathrm{T}}(P, App)$.
4. P *is* multi-path compatible *with App if there is a* $s_0 \in STATES_0$ *such that for every finite command sequence* $\sigma \in P$ *starting in* s_0 *there is a* $\tau \in Sigma$ *such that* $\sigma \cdot \tau \in P \wedge \sigma \cdot \tau$ *executes App. We denote this property by* $Compat_{\mathrm{M}}(P, App)$.
5. P *is* machine-closed compatible *with App if for every finite command sequence* $\sigma \in P$ *there is a* $\tau \in \Sigma$ *such that* $\sigma \cdot \tau \in P$ *and* $\sigma \cdot \tau$ *executes App. We denote this property by* $Compat_{\mathrm{MC}}(P, App)$.
6. P *is* totally multi-path compatible *with App if for each* $s_0 \in STATES_0$ *there is a* $\sigma \in P$ *starting in* s_0*, and for every finite command sequence* $\sigma \in P$ *there is a* $\tau \in \Sigma$ *such that* $\sigma \cdot \tau \in P$ *and* $\sigma \cdot \tau$ *executes App. We denote this property by* $Compat_{\mathrm{TM}}(P, App)$.

By definition,

$$Compat_{\mathrm{TM}}(P, App) \Longrightarrow Compat_{\mathrm{MC}}(P, App) \Longrightarrow Compat_{\mathrm{M}}(P, App);$$

$$Compat_{\mathrm{T}}(P, App) \Longrightarrow Compat_{\mathrm{S}}(P, App) \Longrightarrow Compat(P, App);$$

$$Compat_{\mathrm{TM}}(P, App) \Longrightarrow Compat_{\mathrm{T}}(P, App);$$

$$Compat_{\mathrm{MC}}(P, App) \Longrightarrow Compat_{\mathrm{S}}(P, App);$$

$$Compat_{\mathrm{M}}(P, App) \Longrightarrow Compat(P, App).$$

Compatibility is a non-trivial property. The following example shows that even the weakest form of compatibility, $Compat(P, App)$, may not always hold.

Example 1. Let $App = [\{obj\}, \{op_1, op_2\}, plan]$ with $plan = \{(obj, op_1), (obj, op_2)\}$. Let P be the following property:

"u is the only user who may execute *App*, and a user may not perform all operations of *App* on the same object" (viz., operational separation of duty policies in [5]).

Clearly, P is incompatible with *App* because *App* is not executable in P.

Note that $Compat_{\mathrm{TM}}(P, App)$ and $Compat_{\mathrm{MC}}(P, App)$ imply that the pair of properties (P, App) is *machine-closed* in the sense of Abadi and Lamport [1] and, hence, are properties in the sense of Alpern and Schneider [2]. However, properties $Compat_{\mathrm{TM}}(P, App)$, $Compat_{\mathrm{MC}}(P, App)$, and $Compat_{\mathrm{M}}(P, App)$ are unrealistic. The following example shows that policy properties exist that are *compatible* but not *multi-path compatible* (and therefore neither machine-closed, nor totally multi-path compatible) with the application for which they are defined.

Example 2. Let $App = [\{obj\}, \{op_1, op_2\}, plan]$ with $plan = \{(obj, op_1), (obj, op_2)\}$, and P be the following safety property:

"u_1 and u_2 are the only users who may execute *App*, and a user may not execute two distinct operations of *App* on the same object of *App*" (viz., object- and history-based separation of duty policies [5]).

(1) P is compatible with *App*.

Let s_0 be a start state such that user u_1 has permission op_1 on obj and user u_2 has permissions op_1, op_2 on obj. *App* is executed by the command sequence $op_1(s_0, S_1, obj, s_1) \cdot op_2(s_1, S_2, obj, s_2) \in P$, where S_1 and S_2 are subjects executing on behalf of u_1 and u_2, respectively. However,

(2) P is not totally multi-path compatible with *App*.

If $s = op_1(s_0, S'_2, obj, s'_1)$, where S'_2 is a subject executing on behalf of u_2, clearly, σ is in P and cannot be extended to a command sequence in P that executes *App* (u_1 does not have the permission op_2, and u_2 cannot execute op_1 on obj without violating P).

Properties $Compat_{\mathrm{TM}}(P, App)$ and $Compat_{\mathrm{T}}(P, App)$ are also unrealistic because they require that for every starting state $s_0 \in STATES_0$ there is a (finite) command sequence in both P and *App*, for all P and *App*. Also, we consider $Compat_{\mathrm{S}}(P, App)$ impractical because it may require that administrative commands be executed to allow application execution under a policy P. Hence, $Compat(P, App)$ appears to be the only practical choice of compatibility property.

We note that $Compat(P, App)$ (and also $Compat_{\mathrm{M}}(P, App)$, $Compat_{\mathrm{T}}(P, App)$, $Compat_{\mathrm{S}}(P, App)$) and both forms of $Admin(P)$ fall outside of the Alpern-Schneider framework. Since $\mathcal{P} = P \wedge Admin(P) \wedge Compat(P, App)$, *all* security policies $\mathcal{P}$ – not just information-flow properties [7,12] – fall outside of the Alpern-Schneider framework, and thus are not subject to the Abadi-Lamport composition principle [1].

Let $\mathcal{CS}(\mathcal{P})$ be the set of command sequences that satisfy P, if $AdminP \wedge Compat(P, App)$ is true; and $\emptyset$, otherwise.

Definition 3. *Let App_1 and App_2 be two applications of a secure system, and let $\mathcal{P}_1 = P_1 \wedge Admin(P_1) \wedge Compat(P_1, App_1)$, $\mathcal{P}_2 = P_2 \wedge Admin(P_2) \wedge Compat(P_2, App_2)$, where $P_1, P_2 \subseteq \Sigma_0$ are policy properties for applications App_1, App_2. Let $\mathcal{P}_1 \circ \mathcal{P}_2$ be the policy $(P_1 \wedge P_2) \wedge Admin(P_1 \wedge P_2) \wedge Compat(P_1 \wedge P2, App_1 \sqcup App_2)$. We say that $\mathcal{P}_1$ is* composable *with $\mathcal{P}_2$ if and only if $\mathcal{CS}(\mathcal{P}_1 \circ \mathcal{P}_2) \neq \emptyset$ whenever $\mathcal{CS}(\mathcal{P}_1) \neq \emptyset \wedge \mathcal{CS}(\mathcal{P}_2) \neq \emptyset$.*

If $\mathcal{P}_1$ is composable with $\mathcal{P}_2$, then $\mathcal{P}_1 \circ \mathcal{P}_2$ is called the composition *of $\mathcal{P}_1$ with $\mathcal{P}_2$ or the* emerging *policy, and the set of command sequences $\mathcal{CS}(\mathcal{P}_1 \circ \mathcal{P}_2)$ is $\mathcal{CS}(\mathcal{P}_1) \cap \mathcal{CS}(\mathcal{P}_2)$.*

Even under this very simple composition criterion, realistic policies exist that are non-composable [5].

4 Future Research

In this paper we suggest that application-oriented security policies differ from system-level, global security policies in terms of enforcement, composability, and compatibility obligations. We argue that policy properties such as $Admin(P)$ and $Compat(P, App)$ must be included in any a-o policy definition.

Three areas of future research become apparent. First, new composition criteria need to be investigated that are useful in a variety of application architectures (e.g., some of the criteria addressed by McLean [7]). For example, criteria are needed to define the composition of policies for applications that are privileged (in some limited ways) with respect to an underlying global policy; e.g., application servers that are allowed to violate an operating system or network policy to some limited extent to implement an overall application policy. Furthermore, it is also possible that different composition criteria may be used for different a-o policies within the same system. The properties of policy composition under multiple, coexisting criteria deserve further study.

Second, analysis of policy properties that fall outside the safety-liveness domain becomes necessary beyond that of information-flow properties. For example, it is possible that some properties that fall outside the safety-liveness domain can be "approximated" with properties of this domain, thereby allowing available analysis techniques [1] to be used. Should this be the case, however, the problem of approximation suitability arises naturally. (Preliminary evidence indicates that approximations of information-flow properties with "safety" properties of mandatory access control and covert-channel handling do not compose even under simple property conjunction [11]; i.e., the composition of the approximations does not approximate the information-flow property.)

Third, a-o policies pose added challenges of separate policy definition, selection, and administration; e.g., flexible a-o policy selection and administration impose recurrent costs that must be kept under control. These challenges can be mitigated by the development of practical methods and tools for policy definition, administration, and composition.

Acknowledgments

This work was funded in part by the US Department of Commerce, National Oceanographic and Atmospheric Administration, under the SBIR Contract No. 50-DKNB-7-90120. The views expressed herein are those of the authors and do not necessarily reflect the views of the US Department of Commerce or any of its sub-agencies. The first author would like to thank C. Sekar Chandersekaran of IBM Corporation for his interest in, and support of, this work. We also thank John McLean for his helpful comments.

References

1. M. Abadi, L. Lamport: Composing specifications. In J. W. de Bakker, W. P. de Roever, G. Rosenberg (eds.): *Stepwise Refinement of Distributed Systems. Lecture Notes in Computer Science*, Vol. 430, Springer-Verlag, Berlin Heidelberg New York, 1990.
2. B. Alpern, F. Schneider: Defining Liveness. *Information Processing Letters*, vol. 21, no. 4, October 1985, pp. 181–185.
3. *Common Criteria for Information Technology Security Evaluation*, Version 2.0 Draft, GISA, NNCSA, CESG, NIST, NSA, December 1997.
4. *Federal Criteria for Information Technology Security*, Vol. 1, Chapter 3 and Appendix C. Version 1.0, NIST, NSA, December 1992.
5. V. D. Gligor, S. I. Gavrila, and D. Ferraiolo: On the Formal Definition of Separation-of-Duty Policies and their Composition. Proc. of the 1998 IEEE Symp. on Security and Privacy, Oakland, California, May 1998 (to appear).
6. H. M. Hinton and E. S. Lee: The Composability of Policies. Proc. of 2nd ACM Conference on Computer and Communications Security, Fairfax, Virginia, November 1994, pp. 258–281.
7. J. McLean: A general theory of composition for a class of "possibilistic" properties. *IEEE Transactions on Software Engineering*, vol. 22, no. 1, January 1996, pp. 53–66.
8. *Trusted Database Interpretation of the TCSEC*, NCSC-TG-21, Version 1, National Computer Security Center, April 1991.
9. *Trusted Recovery Guideline*, NCSC-TG-022, Version 1, National Computer Security Center, December 1989.
10. W. R. Shockley and R. R. Schell: TCB Subsets for Incremental Evaluation. Proc. of the Third Aerospace Computer Security Conference Orlando, Florida, December 1987, pp. 131–139.
11. *Unified INFOSEC Criteria*, INFOSEC Concepts, Section 3, Dependencies among TCSEC Requirements (unclassified), National Security Agency, 1993.
12. A. Zakinthinos and E. S. Lee: A General Theory of Security Properties. Proc. of 1997 IEEE Symposium on Security and Privacy, Oakland, California, May 1997, pp. 94–102.

Application-Oriented Security Policies and Their Composition
(Transcript of Discussion)

Virgil Gligor

University of Maryland

Basically I seem to come back to this notion of policies once every five years when I remind myself of the logically difficult problem, namely that of determining the role of security administrators, if any, in systems in general. So the question is, can we shoot this guy and work happily everafter, and the answer is generally, I believe although I'm not quite sure, is that it is no. We basically have to have a certain degree of trust in the ability, intentions and motivations of security administrators. The second conclusion, which I think again is a partial conclusion, is that security administrators are really part of policies so in some sense we are examining the question about what should these good guys do as opposed to what the bad guys do. In other words, suppose that we solved the problem of the bad guys, there are no more bad guys in the world, no men in the middle attack, everything is fine, do we get security problems, and the answer is sure, human failures, bad intentions of this dependent component called a security administrator.

My work in the past like role-based access protocol has tried to address some of the issues of the difficulty, not the complexities, but the difficulty of the security administrator's job and more recently I started looking at application oriented policies and these policies are of course different from global policies, in my opinion, in three fundamental ways, out of which one is obvious. Maybe the others are obvious as well.

The first one is that the scope of these policies are clearly local to an application or a set of applications. They are in force locally and by and large they cannot be used or enforced globally. A couple of examples of local policies versus global policies. If you look at separation of duty policies, those are inherently local policies because the integrity invariants, or integrity properties built into them don't have values in the system outside the application that we are talking about. On the other hand if you are looking at information flow policies, these are global policies because to hold anywhere they have to be enforced everywhere, so there is no way to do information flow where we have a component in the system where the policy is not enforced. So scope and enforcement being the first distinct attributes of these policies.

The second distinct attribute is the fact that these policies have to be composed with other application policies even though the composition result may appear to be trivial these applications policies have nothing to do with each other on the top of the system, they are completely separated. Of course that's a form of composition as well, being able to decide that. So there is a mandatory composition obligation that we have, where you don't have global policies, I mean if you have a global policy say in the network then there is absolutely no obligation of composition. There is utility in composing policies, but no obligation.

The third characteristic, which I believe is very important, probably the most important, and this appears more with application oriented policies than with global policies, but it's not unique, it is one of compatibility. If you look back at what happened in the history of protection mechanisms, you notice that everyone, perhaps starting with Jack Dennis and Lamport's paper, worried about protection mechanisms standing in the way of applications. For example, if you look at the protection rings paper, that Salser and Schroeder did in about 1971, they applied the basic law of programming generality by saying that the protection mechanisms that they built have to support libraries that operate in different protection contexts, such as different rings, without any modification. In other words the ring protection mechanism has to be compatible with procedure call return, for example. Other cases of compatibilities: if you look at, for example, the infamous Bell-LaPadula model, there is a property called compatibility and what that says ultimately is that the security level of a child has to dominate that of a parent because otherwise the root would not be close to the lower levels and you could not resolve paths for children. So the system would fail, it would give it false protection violations, and that property was called the compatibility property.

In our own system work, and when I say ours I mean a number of people, we found that the number one reason why mandatory access control policies failed in practise was not that they were too weak or not good enough, the problem was they were too strong and they broke compatibility of applications. In other words you could not run run-of-the-mill applications on the top of them without violating some information flow, which would basically send the users screaming to the systems security administrator. So essentially, in my opinion, this is what kills all these systems that built too good protection mechanics and end up standing in the way of people, that what it means, incompatible applications. So intuitively a protection mechanism/policy is compatible with an application if it doesn't give you this notion of false protection violations, which most of them do. So the third property that I took into application oriented policies is this compatibility property. Once again, the second one was composibility, mandatory composibility, and the first one was locality of enforcement and locality of scope.

With that I sort of explain what I mean by application oriented and how that differs from global policy and then I have to tell you what I mean by policies. Now this is a very tricky thing because of the merging of two different cultures. You get the culture from the computer security area and you get the culture from the programming concepts area, and both of them converge when they start talking about properties. When they start talking about properties, we have to resolve the issue of what properties are we talking about. Is there anything fundamentally different about the properties making up these policies as opposed to the properties that make up any concurrent system, for example? My controversial statement, hopefully reasoned, is that there are fundamental differences between the two. In fact what I'm prepared to argue is that all policies, in particular all application policies, are not properties in the sense of Alfred and Schneider either, therefore they're not subject to the very known Abadi and Lamport composition principle.

This is one of the points that I would like to make and I am about to go on to tell you how to make the point and then I will give you a notion of a very simplistic, simple is a better word, but I hope you will accept simplistic as a derogatory term for what we do in terms of composition.

The notion of system that we have is a standard notion. We have this notion of states which are usually made up of things including subjects, users, operations, objects, roles, operations being in one-to-one correspondence with permissions for simplicity. We do have a notion of state transition, again the standard one state transitions are given by commands or operations which take as parameter the current state, the subject with all the credentials the subject has, one or more objects and of course produces the next state, assuming that things go well, in other words that this operation is authorised. So whenever you see an .... that means that this is authorised under some policy. So we have the notion of traces of command sequences which people have in programming systems.

We also distinguish, but it's not fundamentally important, tranquil commands from non-tranquil commands. A tranquil command is one which does not alter the security attributes of subjects around it, in other words commands that do not modify access control lists, don't create or destroy the objects, don't create or destroy subjects. Those are tranquil commands. Now it's very important to look at non-tranquil commands as well and that's where I argue that most people miss the boat when they talk about policies.

OK, so system is a set of command sequences, with start states, states S_0 in state calculus sub-zero. Now this is also very important. Why? Security systems assume or security policies assume something about the nature of the system they run in, for example, think in terms of the states as being all the consistent states left in the system after you turn it on but after you recover from a crash. Think in terms of the state being given by an execution of *fsck* in Unix or by recovery from a crash in the Cambridge file system.

These are the states that we are considering start states, states that satisfy some sort of a property that's not necessarily related to the particular policy but it's a useful integrity property. This is a subset of all states in the system. this is quite important as you will see in a minute. We have the notion of secure states and secure commands which are those satisfied by, those that satisfy some properties. Properties in here mean essentially predicates on states and state transitions and later you'll see these predicates on command sequences or sets of command sequences which is equivalent. We obviously thought about reachability of those states as the obvious implication and of our secure systems being systems made up of secure state transitions and secure states. And of course we have Ω a set of all command sequences of a secure system.

So with this in mind, when you try not to define what I really mean of our view of an application and this is again a very simple view, and it is deliberately simple because we want to make the point that even with a simple view of an application, application oriented definition and composition is a hard thing, not an obvious thing maybe not hard for some.

So an application in our mind is a set of objects, a set of operations on those objects, and and the plan of execution of the application is very simplistically defined here, it's simply a finite set of pairs of operations and objects. We don't necessarily talk about ordered pairs or impose an order, we don't talk about least privilege operation of an application execution, in other words if we have an application that calls up obligations and the caller has the fewest privileges and the fewest operations of the callee, on and on and on, we don't talk about other forms of execution plans that are much more sophisticated. But we say that the command sequence Σ, remember all the command sequences, it's the sequence of all those operations on objects and various things, so the command sequences Σ, we say that executes an application if for any pair in the plan there is a command in the sequence, so it's a very simple definition and basically we did not complicate it with any other form of ordering, no form of comparency, absolutely nothing of that sophisticated nature and deliberately so and the reason why is even with a simple definition of execution or executability you'll find that some of the properties of these policies that we deal with are neither safety nor liveness properties.

We actually distinguish these properties, which are again are predicates on command sequences or sets of command sequences at this point, we distinguish them, or classify them, in three groups. The first group is that of attribute properties, and basically these attribute properties include things like the lattice property of security and integrity levels, inheritance properties and membership properties and all hierarchies and permissions, and the like. These are sort of type dependent properties, they are dependent on the type of attribute you use in your access control system.

The second type of properties are the much ignored access management properties of a policy and here you have things like properties of distribution and revocation of permissions with things like selective revocation, transitive revocation, independent revocation, control over who can do a review. For example, I can ask what privileges I have to an object but may I cannot ask what Simon's privileges are to the same object. So there is a certain sense of control there or exclusion. I can ask what perhaps, what are all my privileges in the system, but I cannot ask what are all Bill's privileges in the system. Object and subject creation and destruction, there are actually properties here. For example, privilege inheritance properties for object creation, directory creation, and so on, they differ from system to system. BC has one form of object creation privilege inheritance, UNIX has a different type of inheritance. Object encapsulation, all systems have a form of object encapsulation of protected subsystems. Give me an operating system I can point to where the policy of encapsulation is.

In addition to these much ignored, or generally ignored, properties in a policy, you also have access authorisation. In other words when people talk about policies, or policy properties, they usually exclusively focus on these and on some of the access attribute properties, but not on management. The management is a great missing component. And here you have the obvious things, you have the access authorisation properties given by the access authorisation checks. Take the ones in Bell-LaPadula, Biba, role-based access control, UNIX, on and on and on. This basically tells you that given a subject with a set of credentials, or advocates in this sense, and given an object with a set of attributes, is the access authorised or not. So these are the three types of properties we are interested in policies.

You can partition properties in different ways. The reason why we partition it this way, is to make the first fundamental point that individual security properties, do not make a policy and the reason why they don't make a policy is because there are dependencies among these properties. For example, if you as most people do, take access authorisation properties, like high cannot write to low, and you take attribute properties, something about the liveness properties of security levels, you have not defined a policy, but most Oakland papers say that that's a policy, you actually define some properties of a policy. The reason for this is that if I specify this, and access authorisation depends on access management, I am leaving free variables here and I no longer have a policy.

So, the first point is be very suspicious of people who tell you that they define a policy by defining access authorisation only. In fact I claim that there is no such thing, a policy is more than that. Now with this in mind, the properties that define these three things I call P, and this embodies predicates about these three types of properties and, by and large, these properties are safety properties. In other words they determine that certain bad things don't happen, OK, so most of these published properties in this area are in fact safety properties. Well that's a good thing because we know that all safety properties can be represented as finite automata and we can use finite automata reasoning for them.

They also have another interesting characteristic, maybe they can be enforced by mechanisms outside the system itself, outside the application itself. Unlike, for example, information flow. To do any information flow policy you actually have to examine the code of the particular system, you cannot simply say I am using Bell-LaPadula access checks with subjects and objects and I have an information flow policy. You actually don't, you have an approximation of an information flow policy. An information flow policy means that the union of this property plus a covert channel analysis. OK, so these properties have the advantage that they can be enforced external to the system, in other words I can encapsulate the system, plug in these properties and make sure that they are enforced.

Now, in addition, to these properties, unfortunately, what people ignore are these two notions. A notion of administration, or if there is such a word, administrability, of these properties P. What that intuitively says is that if you start the system in any state you have sufficient powerful commands in the system to bring the system to a state that satisfies the invariance of the safety properties of P. In other words when you turn on your machine and it's in a totally arbitrary state, you can actually bring the system to something that satisfies P, or a set of Ps better yet, and these compatibility properties, which is a lot more complex than it appears, even with my simplification of executability, this property says that there are going to be no false violations of access which break the application itself.

Now, so, formally the two classes of properties in admin P and compat P, actually it's compat P,ac, there is a mistake on this slide but it doesn't really matter because I think I made it clear there is a dependency there, so essentially we took these two definitions for AC we can choose whichever is suitable, in fact the second one maybe so, for each S in the set of states, any state in the system, there existed S_0 in states such that Ω, which is a trace or a command sequence, starts in S and brings the system in state S, actually it's SR, which is S with the transition to itself as a starting transition because P here is the finiteness property of traces therefore you have to have a tracer. Or better yet, for all S_0 states we actually have such a sequence of administrative commands. Now this is not obvious, I mean that this is not trivial, it may be obvious, but it's not trivial. For example, if you take an all capability system that does not have selective revocation you will find out that such a system is not administrable in the sense that there are states in which you cannot bring the system and, in fact, if you have other policies, a separation of duty policies where you cannot erase the memory of what operation was performed, once again you cannot bring the system to certain states. So this is not a trivial property.

Compatibility is one of six possible definitions, and in the paper you will find four, and I'll tell you in a minute why I mention six, is that there exists a state S_0 in states and a sigma which is this trace in policy P, or in property P, which is still three parts of the policy, starting in S_0, such that Σ executes App, in the sense of execution defined. Although you can define other senses of execution and this compat will have the same property, you will have the same definition.

More intuitively this compat is defined as follows. Or is illustrated not defined. You have an application and the application declares a need for admissions to objects and you look at the application, you look at the codes and see for this application to run properly, it needs the following permissions to the following objects, and of course users may need permission to use this application. The policy that we enforce, between users and applications, and applications themselves, because applications can be users too, that policy needs some permissions. The permissions given by this policy have to be compatible with the permissions required by an application, that's ultimately the notion of compatibility.

Here are the six types that we have. The most restricted of compatibility, which is totally unrealistic as you will see in a minute, is that for any state in state zero, for any initial state, that's a state in which the system is fairly consistent but the policy are not necessarily there. For all start states in $Start_0$, the policy traces, all of them, all the traces are need by this policy have to be traces that execute R. We call this totally multi-path compatible.

A second one would say well, some of those start states are really irrelevant for this policy, the policy imposes additional structure on the start states, therefore we take a subset of these start states and we are now saying that for any finite trace starting in the states which are part of P, we can complete this traces with R, which may be an infinite sequence, such that R is executed, and what you notice here, this is precisely 100% the definition of machine closure that Abadi and Lamport have, and there is nothing here that says different. So, what you notice is that our totally multi-paty property implies machine closed compatibility. Also they imply what we call multi-path compatibility. This says that not for all states in P but for some of the states in P, you start all the finite sequences from this subset and you can complete this finite sequences with R. This is turns out is not only not machine closure, but this is not a liveness of a safety property, whereas this is, and this is. As a matter of fact it turns out that out of all this compatibility properties, just like in all the admin P properties, four are not safety and liveness. So in other words, the totally compatible definitions, totally compatible and compatible, which is the one which we choose. These two are in the safety and liveness framework.

So now the question is what choice of compatibility do we have? Well it turns out that we eliminate some of these choices, for example, we eliminate these two choices of compatibility on the basis that they require that all the states which are in state zero have to be initial states in both P, which is the policy traces, and F, which is application. Now this is totally unrealistic. Why? Because P are application oriented policies that actually take subsets of these things. We also eliminate this set of properties, including machine closure because what they say is that for any of these traces that started in all states, or some of the states, or in one state, all these traces, you satisfy F and we already know that that's not the case in most applications. For example, let me just give you one example of this. If you have an application defined by operations op_1 and op_2 on an object, and we have a plan that says operation one operates on objects and operation two operates on objects, and P is the property that U1 and U2 are the only users who may execute App, and the user may not execute two distinct operations on the same object, such as an operation of separation of duty, defined by Kuhn, Feranault and Fugini, then we have some traces for which compat is true and some other traces, starting in S_0, same state, for which compat is not true. So it's a trivial example, but this point the lack of a realistic, even of multipath compatible even when he says there exists some states, not all states, out of which all this finite sequences followed by τ, which is a maintenance sequence exhibitor. So even in something as trivial as this, a trivial property and a trivial system definition, this doesn't hold. You can imagine what happens for these properties that are wider, in other for machine closure and for the restricted machine closure if you are in the sense of Abadi and Lamport.

So the upshot of this is we could eliminate out of our discussion of compatibility for both theoretical and practical reasons, just about everything except these two notions of compatibility. This is the most general. It says that there exists a sequence, that's part of P, which executed *App*. Again this is neither a safety nor liveness property. Now what about a strongly compatible property. When we eliminate this, it's not that it's a disaster but notice what happens. To start any application, I have to bring the system to admin P to this state, OK, and furthermore from this state I have to do more work, in other words, to initiate any application I have to have an administrator sitting by the side, initial some commands to generate this sequence that executes *App*. Now that's obviously impractical rather than complex so, for that reason, we chose this notion of compatibility. You can choose any notion if you want out of here but be aware that two in the safety and liveness framework and the other four are not.

So the basic point is that if this is the case, and I have a bunch of examples which show that this composition, the simplest possible composition criteria that you could have, tells you that two applications may not necessarily be composable because they are not compatible or because they don't have enough commands in admin P to administer the system. The definition of the emerging policy and public criteria is very simply the one that has two and Heather Hinton had, and by the way, they didn't notice that these properties here have to be non-conflicting because otherwise the policy is zero. So if you don't have admin P the policy is zero. So if you don't have compat P and *App* the policy is empty not zero. Basically, the criteria says that the policy P_1 intersect the policy P_2 is non-empty whenever P_1 and P_2 are non-empty and the emerging policy is an intersection, or a conjunction, of these type properties, management properties, and access authorisation properties, with their administrability property and their compatibility property.

So the point that I am trying to make, the fundamental point is clearly the systems have to be administrable in some sense, they have to have an administrator that has to be able to define, install and initialise privileges for all applications and that administrator, the action of that administrator, is actually part of the policy itself. When you look at P made up of those properties that is basically a policy template. So when people talk in, for example the Oakland Conference about policy, this policy or that, not only they give you only part of it but when they give you the whole thing which is very rare and they talk about the template and ignore what the administrator does and how they administer the system. So, basically what this says is that the whole theory based on safety and liveness, I argue it is inappropriate here, it just doesn't work, we have to do other kinds of analysis for policies. If, on the other hand, you want to limit yourself to individual properties, then safety and liveness may be OK, I mean it's a great area of research to take individual properties and determine their safety and liveness and the composition of individual properties, that's useful, but it's not about policies, and I hope that that's reasonably clear of controversy. Of course not everybody accepts that.

Stewart Lee: There have been two proofs recently in the literature, that all security properties are not Alfred and Schneider.

Reply: No, only information flow policies I believe ..

Stewart Lee: You're absolutely right, I misquote, you're absolutely correct.

Reply: I don't consider them interesting anymore, because they all break compatibility.

Secure Fingerprinting Using Public-Key Cryptography
(Position Paper)

Hiroshi Yoshiura, Ryoichi Sasaki, and Kazuo Takaragi

Systems Development Laboratory, Hitachi, Ltd.
292 Yoshida-cho, Totsuka-ku, Yokohama, 244-0817 Japan
e-mail: {last name}@sdl.hitachi.co.jp

Abstract. Fingerprinting is a process that embeds identifiers of the buyers of data into the data. It enables buyers who copied and redistributed data illegally to be identified from the redistributed data. An essential requirement for fingerprinting is the prevention of false accusations, i.e., honest buyers not be accused even when fraud has been committed by merchants and third parties. Previous fingerprinting methods either could not meet this requirement or met it at a high cost, such as that associated with the use of independent servers for fingerprinting.
This paper proposes to embed buyers' digital signatures into data and to identify illegal buyers by verifying signatures in the redistributed data. The security of the signature verification is discussed assuming that the redistributed data have been modified by the illegal buyers. The paper shows that the proposed method can prevent false accusations at an acceptable cost.

1 Introduction

Fingerprinting is a process that embeds identifiers of the buyers of data into the data. It enables buyers who copied and redistributed data illegally to be identified by extracting their identifiers from the redistributed data.

Although reliable identification of illegal buyers is an essential requirement for fingerprinting, previous fingerprinting methods have not been sufficiently reliable when merchants and third parties have committed fraud [1,2,3,4]. This paper proposes a fingerprinting method that solves this problem by using a public-key cryptographic protocol.

Section 2 clarifies requirements for secure fingerprinting. Section 3 surveys previous study and clarifies problems in this field. Section 4 describes the new method, Sect. 5 discusses the security of the proposed method, and Sect. 6 concludes the paper.

2 Requirements for Secure Fingerprinting

We are applying fingerprinting to several fields, such as the sales of land photographs taken from artificial satellites. The purposes of fingerprinting in these fields are the following:

1. Deter buyers from illegal copying by informing them that merchants can identify the illegal buyers.
2. Prevent increased illegal copying by suspending sales to the detected illegal buyers.

To realize these purposes, the following requirements need to be satisfied:

1. High Detection Ratio
   The ratio of the number of illegal buyers detected to the total number of illegal buyers must be sufficiently high for practical use.
2. Prevention of False Accusations
   The ratio of honest buyers incorrectly determined to be illegal buyers must be negligible small.

The prevention of false accusations is more critical than assuring a high detection ratio. That is, fingerprinting might be still useful even if it overlooked 50 percents of illegal copying but would be useless if it accused 1 percent of the honest buyers. There have, however, been few studies of prevention of false accusations [2,3,4] even though the studies of assuring a high detection ratio [5,6,7,8,9] have been active. This paper therefore focuses on the prevention of false accusations.

3 Previous Fingerprinting Methods

This section clarifies problems with previous methods by analysing four representative methods. We first define some terminology.

ID identifier of a buyer of data.
ENC(PD, SEK) data obtained by encrypting plain data PD using function ENC with secret encryption key SEK.
OWF(D) data obtained by applying one-way function OWF to data D.

Now we analyse the previous methods. With an earlier method [1], merchants embedded buyer IDs into data and identified illegal buyers by extracting IDs from redistributed data. This method had the following false-accusation problems:

Frauds by Merchants Honest buyers could be accused when merchants illegally copied and redistributed data with embedded buyer IDs.
Frauds by Third Parties Honest buyers could be also accused when third parties change IDs in data and redistribute the data illegally.

Method [2] solved the first problem, the one due to merchant fraud, by hiding buyer IDs from the merchants. It used the following procedures:

1. The buyer generates his ID.

2. The buyer applies a one-way function OWF to the ID, and sends $OWF\,(ID)$ to the merchant. The merchant sends data to the buyer. The buyer embeds the ID in the data.
3. When the merchant captures an illegally copied image, he extracts ID' from that data and compares $OWF\,(ID')$ with $OWF\,(ID)$ received in 2 to identify the illegal buyer.
4. The merchant sends the data and records obtained in 3 to the arbiter in order to accuse the illegal buyer.

This method had a problem, however, in that IDs were independent of the data. Thus, once the IDs came to be known by other people, the IDs could be embedded in any other data. Honest buyers could be accused when other data with their IDs were redistributed illegally. Thus this method did not solve problem 2 of the earlier method, the one due to third party fraud.

Method [3] introduced embedding servers who received data from merchants and embedded buyer IDs in this data independently of the merchants. Thus merchant's frauds were prevented by putting the embedding servers between the merchants and the buyers. This method, however, did not solve problem 2, and it increased the costs of system operation because embedding servers had to be used whenever the merchants sold data to the buyers.

Method [4] let buyers secretly select parts of data. The method had the following procedure in selling phase:

1. The merchant divides data into blocks $B_1, B_2, \cdots, B_\mathrm{n}$ and generates m copies for each of $B_i\,(i = 1, \cdots, n)$. Information is embedded into each of these copies $B_{i,k}\,(k = 1, \cdots, m)$ to discriminate between them. These $n * m$ copies are encrypted and the resultant $ENC\,(B_{i,k}, SEK_\mathrm{m})\,(i = 1, \cdots, n; k = 1, \cdots, m)$ are sent to the buyer.
2. For each $i\,(i = 1, \cdots, n)$, the buyer selects a copy $ENC\left(B_{i,s[i]}, SEK_\mathrm{m}\right)$ from $ENC\,(B_{i,k}, SEK_\mathrm{m})\,(k = 1, \cdots, m)$. The selected copies are encrypted, and the resultant $ENC'\left(ENC\left(B_{i,s[i]}, SEK_\mathrm{m}\right), SEK_\mathrm{b}\right)\,(i = 1, \cdots, n)$ are sent to the merchant.
3. The merchant decrypts the received blocks and sends the results to the buyer. (ENC and ENC' are commutative).
4. The buyer decrypts the received data in order to obtain the plain data.

An honest buyer is safe from the fraud of both the merchant and the third-party fraud because they could not know what copies the buyer had selected and thus could not redistribute the same data that the buyer had received. This method, however, requires a three-path communication in the selling phase - (steps 1, 2, and 3) - and, furthermore, the first path sends m copies of data. Thus the communication costs are much higher than those of the earlier method. The method also has another problem: the buyer's secret key SEK_b used in step 3 are needed when an arbiter decided whether or not the buyer is illegal. Thus the buyer has the burden of recording SEK_b correctly.

Table 1 summarises the analyses in this section. It shows that the earlier method [1] had two problems: frauds by merchants and fraud by third parties.

The succeeding methods [2,3,4] either could not solve these problems or else solved them at a high cost. Thus none of the previous methods could prevent false accusations at a reasonable cost.

Table 1. Analyses of Previous Methods.

Methods	Source of Attack		Cost
	Merchant	Third party	
Hiding *ID*s from merchant [2]	Solve	-	None
Introduce embedding server [3]	Solve	-	Operation cost of embedding server
Buyer's secret selection [4]	Solve	Solve	- Increased communications between merchant and buyer - buyer needs secret keys for denial

4 Secure Fingerprinting Using Public-Key Cryptography

This section proposes a method for solving the two problems mentioned in Sect. 3 at a cost that is practically acceptable.

4.1 Basic Proposals

An obvious solution to the problem of merchant fraud is to hide buyer *ID*s from the merchants, but the merchants need to recognise what buyers the *ID*s represent. This asymmetric requirement suggests the use of public-key cryptography.

Preventing the frauds by third parties requires correspondence between data and the embedded information. A reliable way to establish this correspondence is to use hash values of data in the embedded information.

These considerations led us to the use of buyers' digital signatures for data as embedded information. Thus our basic proposals are the following:

1. Embed the buyers' digital signatures for data into the data.
2. Identify illegal buyers by verifying the digital signatures in the redistributed data by using buyers' public keys.

4.2 Signature Verification in Fingerprinting

Illegal buyers often use part of data or modify the data before redistribution. They may, for example, cut out a part of an image or change its brightness. Thus the redistributed data might not be identical to the original data. The signature verification procedure in our method thus has to cope with the following problems:

1. Hash values of the redistributed data are different from those of the original data.
2. Digital signatures embedded in the redistributed data have been partially lost.

We solve problem 1 by using hash values of the original data instead of using those of the redistributed data. And we solve problem 2 by using error correction codes to correct digital signatures. Figure 1 illustrates the signature verification procedure in the proposed method.

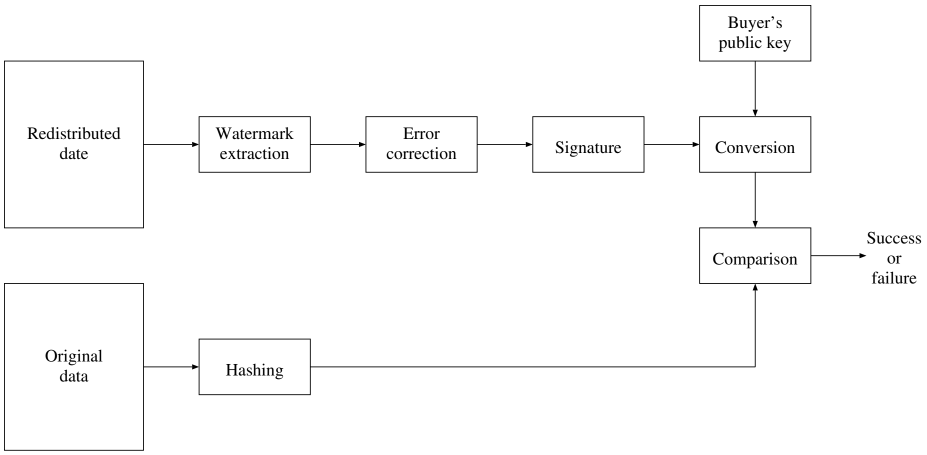

Fig. 1. Signature Verification in the Proposed Method

Another problem in the signature verification is that buyers may not distribute correct public keys. The solution is that the merchants encrypt data by the buyers' public keys when sending them to the buyers. If the buyers' public keys are not correct, the buyers could not decrypt data by using corresponding secret keys.

4.3 The Protocol

The protocol of the proposed method is illustrated in Fig. 2. A problem with this protocol, however, is that it allows buyers to tamper with the system and thereby obtain data without embedding signatures. We therefore need to make the system tamper-resistant by using tamper-resistant software [10] or hardware such as smart cards.

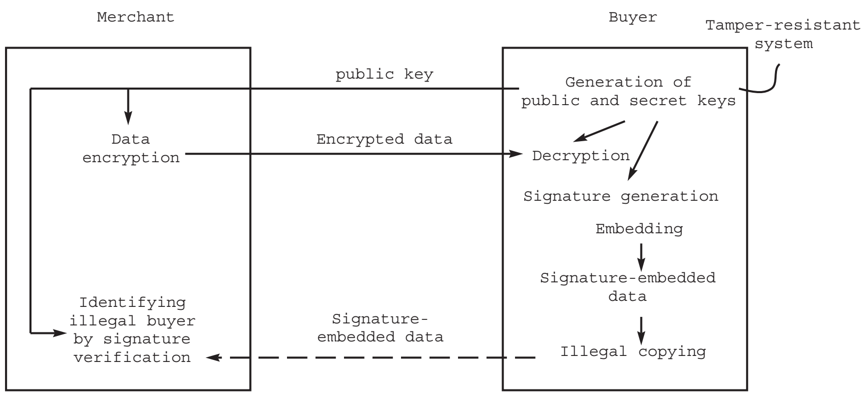

Fig. 2. Protocol of the Proposed Method

5 Security of the Proposed Method

The security of the proposed method can be reduced to that of the underlying signature scheme when the redistributed data has not been modified by illegal buyers. We thus discuss the case of the modified redistributed data.

In Sect. 4.2, we proposed two solutions to the signature verification for the modified redistributed data, i.e., using hash values of the original data and using error correction codes. The influences of these solutions on the security of the signature scheme is discussed below.

1. Influence of Using Hash Values of the Original Data
   There could be two types of fraud by which honest buyers are accused, forge of the signatures by merchants and third parties and fraud by the merchants in the signature verification. We first consider the forge of signatures. The possibility of the signature forge is irrelevant to whether the original data or the redistributed data is hashed in the verification phase. It only depends on whether or not the buyer's secret keys are known to the forger and depends on the number of bits in a signature. The risk of the signature forge is thus the same as that of the underlying signature scheme.
   We next consider the merchant fraud in the signature verification. Hashing the original data instead of the redistributed data allows the merchants to take another data and insist that this data is the original one. However, the hash value of the new data has to coincident with the signature for the deceiving purpose. Thus the risk of the merchant fraud is that of the merchant's finding an inverse value of the hash function, which is the same as the risk of the underlying signature scheme.

2. Influence of Using Error Correction Codes

   It is obvious that signatures corrected by error correction codes have the same level of security as the original ones.

In summary, the security of the proposed method is reduced to that of the underlying signature scheme.

6 Conclusion

The prevention of false accusations of fraud is critically important for secure fingerprinting. This paper clarified problems in meeting this requirement by analysing previous methods and proposed a new method using digital signatures. The paper also discussed the security of the signature verification in fingerprinting environments, and it showed that the proposed method can prevent false accusations at a cost that is practically acceptable.

References

1. N. R. Wagner, *Fingerprinting*, Proceedings of the 1983 Symposium on Security and Privacy, IEEE Computer Society, April, 1983, pp. 18-22.
2. B. Pfitzmann and M. Schunter, *Asymmetric Fingerprinting*, Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin, 1996, pp.84-95.
3. N. Miura, H. Watanabe, and T. Kasami, *A Strong Watermarking against Server's Corruption*, Proceedings of the 1997 Symposium on Cryptography and Information Security, Fukuoka, Japan, January, 1997.
4. K. Baba, K. Iwamura, Z. Yuliang, and H. Imai, *An Interactive Protocol for Image Fingerprinting*, Proceedings of the 1998 Symposium on Cryptography and Information Security, Hamanako, Japan, January, 1998.
5. G.R. Blakley, C. Meadows, and G.B Purdy, *Fingerprinting Long Forgiving Message*, Crypto '85, LNCS 218, Springer-Verlag, Berlin, 1986, pp. 180-189.
6. D. Boneh and J. Shaw, *Collusion-Secure Fingerprinting for Digital Data*, Crypto '95, LNCS 963, Springer-Verlag, Berlin, 1995, pp. 452-465.
7. I. J. Cox, J. Kilian, T.Leighton, and T. Shamoon, *A Secure, Robust Watermark for Multimedia*, Proceedings of the Fist International Workshop on Information Hiding, LNCS 1174, Springer-Verlag, Berlin, 1996, pp. 185-206.
8. H. Watanabe and T. Kasami, *A Secure Code for Recipient Watermarking against Conspiracy Attacks by All Users*, Proceedings of Information and Communications Security, LNCS 1334, Springer-Verlag, Berlin, pp. 414-423.
9. J. Yoshida, K. Iwamura, and H. Imai, *A Coding Method for Collusion-Secure Watermark and Less Decline*, Proceedings of the 1998 Symposium on Cryptography and Information Security, Hamanako, Japan, January, 1998.
10. D. Aucsmith, *Tamper Resistant Software*, Proceedings of the Fist International Workshop on Information Hiding, LNCS 1174, Springer-Verlag, Berlin, 1996, pp. 317-333.

Secure Fingerprinting Using Public-Key Cryptography
(Transcript of Discussion)

Rojoichi Sasaki

Hitachi

I am Rojoichi Sasaki, from the Hitachi Systems Development Laboratory in Japan. Today I will talk about secure fingerprinting using public key cryptography.

First of all I will explain the requirements for secure fingerprinting. Fingerprinting is a technology for copyright protection of digital data and it is an application of digital watermarking. The function of fingerprinting works as follows.

First it embeds the buyer's id, in an invisible manner, in the data, i.e. still pictures, motion pictures, etc. Second it identifies the buyers who copy and redistribute illegally the data they legitimately bought.

Let us suppose a buyer that illegally sells copies of data he legally bought. The role of fingerprinting methods in such a situation is first to deter buyers from illegal copying by informing them that merchants can identify the illegal buyers. Second to prevent an increase of illegal copies by suspending sales to the detected illegal buyers.

These goals must be achieved satisfying at the same time two other requirements. First, high detection ratio, defined as the number of illegal buyers detected divided by total number of illegal buyers. This ratio is desirable to be high. Second, prevention of false accusations, defined as the number of honest buyers incorrectly accused, divided by total number of buyers. This measure is more critical than the first ratio because we doubt about our customers, therefore, this ratio must be very, very small.

Next I will show the list of the fingerprinting methods that have been proposed in literature. I will explain in more detail the method proposed by Wagner in 1983.

This method considers two processes. One process sells the data to the buyer, the other process identifies the illegal buyers. In Wagner's method a merchant generates a buyer id and this id is an integral part of the data. The data sold by the merchant contains always the buyer's id. The illegal distributed data can be captured by the merchant and the merchant can extract the original id and identify the illegal buyer. However, in this method an honest buyer could be accused in two cases. First, when the merchant illegally distributes the same data already legally sold. This case occurs only when an employee of the merchant's company does such a thing. Second when the merchant, or some other party, illegally redistributes other more expensive data with the same id, this is possible in Wagner's method because the data and the buyer id are independent.

We propose fingerprinting based on digital signatures. The buyer first generates a pair of public and secret keys, and sends the public keys to the merchant. When the merchant sells the data, the data will be encrypted by using the received public key. The buyer decrypts the data and performs the signature generation and then he embeds it into the data as a fingerprint. Every copy of the data can be captured by the merchant and the merchant can identify the illegal buyer by signature verification based on the received public key.

We will discuss the accessibility of the proposed method from three viewpoints. One is false accusations and frauds by merchants and other parties. Second detection of illegal buyers, here frauds by buyers are the problem. Third is reliability of the signature verification.

As mentioned earlier in the previous methods an honest buyer could be accused in two cases. Therefore, we propose to use the buyer's data signature on data as the buyer's id. By using this method the merchant cannot generate the buyer's signature because the merchant does not know the buyer's secret key, therefore, the first problem can be solved. In this method data and id are inseparable because we use the data signature and the data signature authenticates the data then, also the second problem can be solved.

Next, we discuss the detection of illegal buyers. We have two types of attack by illegal buyers. The first attack consists of sending a false public key to the merchant. To solve this problem we propose to use the same key for data encryption and the signature verification as displayed here. If the buyer sends the false public key to the merchant, the merchant will encrypt the data with the false public key. In this case the buyer cannot decrypt the data correctly, this method forces the buyer to send the correct key to the merchant. The second attack consists of obtaining the data before fingerprint, just after decryption. To solve this problem we propose the use of tamper resistant software or hardware.

Next I analyse the reliability of the signature verification. The illegal buyer uses part of the data or modified data before distribution, for example, the illegal buyer cuts off the e-mail data or changes the scale of the e-mail data, therefore, we have the following two problems. First the digital signature of the data has been damaged because of the partial loss of the data. To solve this problem we use error correction code. Second the value of the hash function does not match with the distributed data. To solve this problem we use the original data as input of the hash function.

In some cases, the error correction code fails to correct the data because of a heavy loss of the information. However, this problem is present also in the other methods of fingerprinting, therefore I think that the proposed method is acceptable from a security and reliability point of view.

We conclude saying that this method could be a base for the future study of fingerprinting.

Peter Landrock: You have to embed a signature into the picture and how is that embedded in such a way that it cannot be removed?

Reply: This problem exists. Is the same in the old natural fingerprinting or a watermarking method. We adapt the method so that it is difficult to remove.

Anyway the same problem is present in other fingerprinting or watermarking methods.

Virgil Gligor: You mentioned, maybe I am mistaken, but you are using the same key for signatures and encryption. Generally I am told by my crypto brethren that it is not a good idea to use the same key for two separate functions, and there are some standard examples why you wouldn't use a signature key for authentication. Did you find any problem in doing the cryptographic analysis of this with the fact that you are using the same key for two separate purposes.

Reply: In my opinion, there are no big problems to use the same key in this case.

Virgil Gligor: We know that there is a big problem for signatures and authentication, but maybe not in this case.

Peter Landrock: I would support that, I certainly think you can develop a protocol where you do not run into problems.

Angelos Keromytis: I have a question about the overall architecture of your scheme, you showed that the piece of software that requires PC, that was labelled trusted, tamper-proof software. How realistic is that? If something's running on my PC why wouldn't I be able to just change it and do whatever I want?

Carl Ellison: How can you achieve tamper resistance ?

Reply: It is possible in two ways. First by using tamper resistant software made by using a special compiler, even if I think this is not such a good idea. Second, this process may be performed for example by a smart card. In this case it is possible to have tamperproofness but the cost is increased.

Joan Feigenbaum: So it is tamper resistant hardware and not software.

Stewart Lee: You have converted tamper resistant software into tamper resistant hardware. So accepting that there is not such a thing as a tamper resistant smartcard.

Francesco Bergadano: That seems to be the main weakness of the approach, the need of tamper resistant hardware, and I was wondering if the problem that you address is actually the possible fraud by merchants as compared with previous work mainly concerned with fraud by buyers, and the work by Pfitzmann, presented at the ACM Security Conference last year pressed the same problem but did not need the tamper resistant hardware. They had a rather more complicated protocol based on zero knowledge proofs, so why are you using this that requires the tamper resistant and not Pfitzmann approach.

Reply: In some cases this type of system would be acceptable because, as I mentioned earlier, we have two requirements, one requirement is to reduce false accusations, in this case a buyer can avoid the hardship, but it did not solve the big problem, so I think this method is not perfect, but in some cases it is acceptable.

Joan Feigenbaum: This is actually a question to anybody who has an answer for it. I have seen a lot of talks lately about watermarking.

Reply: I use fingerprinting as a special case of watermarking.

Joan Feigenbaum: So, the overall picture is that you use watermarking or fingerprinting to catch thieves, how successful has this been, I guess I am a little sceptical, just as I am a little sceptical about tamper resistant software, I am really sceptical that's it is all that feasible actually to track down these watermarked copies of digital content and find the bad guy, so has this actually been done successfully, with commercially valuable content?

Stewart Lee: No.

Peter Landrock: I think it has got to be said that this is a very old discipline, I mean the world used to be sceptical about public keys.

Angelos Keromytis: The big problem with watermarking is that it may work once, but once there is a trial or something, then the company that does the watermarking has to show to everyone's satisfaction that this thing is really watermarked and this guy wasn't supposed to have it, then everyone knows the method and essentially from that point everyone can just forge or just remove it.

Stewart Lee: One of the big problems with watermarking is that typically in a watermark you change a few pixels here and there according to some algorithm. Then a villain can destroy the watermark merely by distorting the picture, only slightly, change a rectangular picture into a trapezoidal picture.

Joan Feigenbaum: OK, I think you guys are talking about something different.

Roger Needham: It is worth remarking the computer lab folks who do this are not here because they are information hiding, and they have a utility that removes the watermark from anything, at least it does not remove the watermark, it prevents the programme that is written to find it, from finding it, which is a different matter from removing it.

Joan Feigenbaum: So now you have actually identified three different technical failures of the watermarking system, I believe in all those plus more actually, like linearity, for example, you haven't mentioned that, but I guess I'm sceptical that you can even really track down these stolen copies. I am very sceptical about it.

Stewart Lee: You have to have an idea that a copy is perhaps stolen and then you can examine it for all of them.

Joan Feigenbaum: The principle that is always articulated when I hear this is, ok if it is just a few copies we do not care. If it is a large number of copies it will just be obvious because it is a large number of copies. I believe the first, I am just not so sure this smooth trade off between the level which we do not care about this copying because it's not commercially significant and the level at which we do care and furthermore can catch them, it is the whole systemic issue, you have to catch them before you can run your watermark.

Stewart Lee: There are many aspects to catching and one of the things you can do is you can prevent legal players from exhibiting the thing that is fingerprinted or watermarked and consequently it will only be exhibitable on illegal players which are much more easy to catch.

Michael Roe: There is work being done on prevention as well as detection where you do not just try and catch people with illegal copies you just make sure that they just cannot actually interpret it and from a technical point of view it is harder but it is much more useful if you can make it work.

Joan Feigenbaum: It involves this secure software issue.

Angelos Keromytis: No actually, it is tamper resistant hardware.

Joan Feigenbaum: OK, well that is different, I believe in that.

Virgil Gligor: For example, hiding bits in a picture, which is called steganography, basically you can hide the watermarks such that it is below probability that somebody can find it other than the checker. Unfortunately with very high probability that other people can remove it, even if they cannot find it.

Stewart Lee: Yes. You can remove enough bits so that it does not make sense, that is true.

Peter Landrock: Well, it is still a young subject of course, but then we should be grateful that the fact that it is not practically applicable yet does not stop people from doing research on it. Actually there was a very interesting talk at the Neutron Conference on secure protocols where you perform the Fourier transform and then you did your watermarking with a Fourier transform and then you transform back again, but you cannot automatically Fourier transform without having the right keys. So at least I certainly think there are some interesting possibilities although you have a good point there.

Third Party Certification of HTTP Service Access Statistics[⋆]
(Position Paper)

F. Bergadano and P. De Mauro

Dipartimento di Informatica, Università di Torino, Italy
bergadan@di.unito.it - http://maga.di.unito.it

1 Introduction and Previous Work

Online advertisement represents a significant portion of Internet-related revenues. However, no reliable and widespread system for metering Web site popularity has emerged. As a consequence, the market is dominated by the few very well-known sites, such as the main newspapers and the most effective search engines, because their popularity is taken as a given fact. For many sites of medium to high popularity there is a great potential for an increase of advertisement profits, but third party guaranties for their declared access rates is needed. There has been surprisingly little work on Web metering, and most of this work is not concerned with certified statistics. In fact, Web counting is a difficult and complex problem, raising many issues of different kinds. Even when the institution running the Web server is trusted, the obtained statistics may not correspond to real usage. Some traffic is generated from within the local area network, mainly for the site's development and testing, and should be eliminated. Some other traffic is generated by Web spiders and other automated search engines. This should also be treated differently. A large number of external user accesses are hidden by proxies at different network locations, and it would be interesting to quantify the number of such additional hits. An in depth analysis of these issues is found in [6].

From a security perspective, though, we are especially interested in avoiding metering data falsification by the Web site's institution. There have been a number of recent commercial initiatives aimed at WWW counting certification (Nielsen/IPR, Netcount, Audinet), but the success and the actual security of such projects will not immediately become clear. In the security literature, there have been two different studies in this area: the first [1] addresses the problem of statistics being falsified at the server's site, the second [2] addresses the problem of artificial requests generated by clients just for increasing the number of server hits. Naor and Pinkas [1] base their work on a secret sharing scheme. A simplification of their method is as follows: a secret is distributed by the audit

⋆ The techniques described herein are subject to patent protection (clearance request sent to the Italian Patent and Trademarks Office, November 1998).

agency to all N clients, and the server needs to receive requests from at least K clients in order to obtain the secret. The secret is then sent by the Web server to the audit agency as a proof of having received at least K hits during the audit period. The main drawback of this proposal consists in the fact that clients need to be initialized for each audit period, and this forces a communication with the audit agency. We believe this requirement makes the scheme not practical, because Web users will not be willing to do extra work and wait extra network time because of somebody else's metering needs. Franklin and Malkhi [2] address the more difficult problem of artificial client requests. Their solution is based on a timing protocol that makes a large number of artificial requests very costly. Every time a client does an HTTP request, this is forced through an auditing proxy, that keeps some logs, and appends a special timing applet to the Web server's response. The timing applet will run on the client's computer while the corresponding page is being viewed, then the result of the computation is sent back to the auditing proxy. The verification of this output by the auditing agency is only probabilistic and reasonably fast. The main drawback of this approach consists in the computation time that clients are forced to devote to this purpose. The server's popularity may be damaged by this client cpu time waste, more than it is increased by certified metering.

2 Proposed Setting and Solutions

We address the problem of metering falsification at the Web site, either by direct modification of logs or by issuing artificial HTTP requests with spoofed IP addresses. We require the WWW server to be equipped with a special tamper-proof hardware that is provided by the certification agency. The hardware has a simple function and is connected directly to the WWW server (e.g., to its parallel port). The site's WWW server will need to be modified, either with software patches provided by the certification agency, or through a separate, independent vendor. The software modification does not affect server performance, except for hits that are randomly selected for verification. This is a very small portion of all the HTTP requests that are received. The certification agency has an online HTTP server, that is used for performing some logging, and receives requests from the clients. Clients are obviously everywhere on the Internet, and they are not modified or contacted previously - their IP addresses are not known beforehand. The is global picture is given in the Figure.

The tamper-proof hardware connected to WWW server is a black box that behaves as follows:
- it receives a line of input.
- it produces a random bit, obtained through a cryptographic generator. The generator needs to be modified, so that the probability of 1 is very low (e.g., 0.01 or less). The bit can only be predicted with a corresponding cryptographic key, that is possessed by the tamper-proof device and by the certification agency, not by the WWW server institution.
- the bit is output to the WWW server.

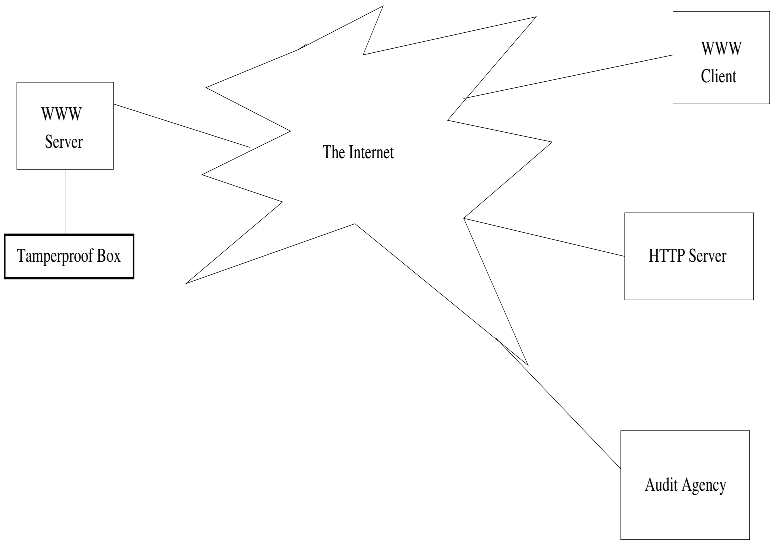

Fig. 1.

- if the bit is 1, the line is marked.
- the line's integrity is protected by performing symmetric authentication (by-computing a MAC, that is stored externally). When log certification is required, the line and its MAC will need to be sent to the audit agency.

Using tamper-proof hardware raises a number of questions and possible criticism, given the recent literature on related attacks (see, e.g., [3,4]). Although such criticism could apply to the present study, and should be carefully considered, it is less problematic than in other contexts. First of all, the WWW site's institution will not devote to tampering more resources than are justified by the value of the statistics produced during one certification period. The situation would be different if the hardware had to protect long term master keys for symmetric cryptography, or the private key of a Certification Authority. Second, the recent Differential Fault Analysis attacks to both asymmetric and symmetric schemes are significantly more difficult if the encryption process is repeated or verified. This is prohibitively inefficient in some applications, but it can be done easily here, because we only need to generate one bit per hit, and the corresponding log data needs to be authenticated; the number of hits per second should not be a problem for conventional encryption, even if it has to be repeated two or three times. Finally, Differential Fault Analysis has not yet been applied to cryptographic random number generation, although it could probably be extended to simple schemes such as the generation of blocks of 64 random bits with DES under OFB mode. More complex schemes such as ANSI X9.17 could be harder to break [5]. We could also generate random numbers based on physical measurements inside the tamperproof box, in case cryptographic random number generation causes tampering problems.

The WWW server's operation must be modified as follows:
- when an HTTP request is received, the corresponding log line, or the part of it that needs certification, is output to the tamper-proof device;
- if the generated bit is 0, the request is serviced normally
- if the bit is 1, the request is serviced with an HTTP response that contains a redirect to the certification agency (the HTTP status is of type 302, and a location header with the address of the certification agency is returned)

Everything else works as normally required by HTTP. When the client receives a redirect response, it will automatically re-issue the request to the certification agency, where a corresponding log line is stored on disk for subsequent checking. The certification agency's WWW server will also issue a second redirect response, back to the original server, so that the request may finally be served with the actual data. This will take a bit longer than a normal response, but it will occur very rarely. Even in those rare cases, though, the client will receive a correct reply with a reasonable delay.

It should also be noted that the scheme avoids the creation of artificial hits by clients with a spoofed IP number, if this IP number is inexistent, or if it belongs to an area in the Internet which is not controlled by the attacker. In fact, the redirect hit to the audit agency's server will be a normal HTTP request, that needs a TCP connection establishment: the corresponding three-way handshake would not work with a spoofed IP number. Before sending the HTTP request to the audit agency, the client must initiate a TCP connection under the spoofed IP number, but the audit agency's response will not get back to the client. Since this response contains the initial TCP sequence number for the audit agency's side (a kind of a nonce), the spoofed client will be unable to produce a consistent reply, and the connection will be closed. Clients will of course be able to produce artificial requests using their own IP numbers, or IP numbers that are controlled by the attacker (e.g. from a local network with only one entry point). The technique proposed in [2] could be used if this kinds of false hits also need to be detected.

The certification agency will accept the log files submitted by the WWW server's organization if all the log lines marked by the tamper-proof device correspond to an associated client hit at the agency's site. If this is not a case in a repeated number of occurrences, certification could be denied, based on the agency's policy.

Acknowledgments
We would like to thank Bruno Crispo and Franco Sirovich for helpful comments and discussions.

References

1. M. Naor and B. Pinkas. Secure Efficient Metering. Eurocrypt 1998.
2. M. K. Franklin and D. Malkhi. Auditable Metering with Lightweight Security. Proceedings of the Financial Cruptography Workshop, 1997.

3. R. Anderson and M. Kuhn. Tamper Resistance a Cautionary Note. Proceedings of the second Usenix Workshop on Electronic Commerce, pages 1-11, Oakland, 1996.
4. E. Biham and A. Shamir. A New Cryptanalytic Attack on DES. Research announcement, available, e.g., at http://www.infowar.com/infosec/infosec.html-ssi and at http://jya.com/dfa.htm .
5. ANSI X9.17 Financial Institution Key Management (Wholesale), American Bankers Association, April 4, 1985.
6. J. E. Pitkow, In Search of Reliable Usage Data on the WWW, Proceedings of the VI Int. WWW Conference, Stanford, 1997

Third Party Certification of HTTP Service Access Statistics
(Transcript of Discussion)

F. Bergadano

University of Turin

My name is Bergadano and this work is co-authored with my students. We are concerned with certifying access rates to servers of some kind and of course the main application we have in mind is web counting. Some web site says they get so many hits a day and we want to have an agency, an independent agency, certifying that it's true.

Web counting, well the method actually is applicable also to other types of protocols including services such as DNS and directory services. We need protocols though that allow for a redirection. Anyway web counting is at the same time a difficult problem and, we believe, a very important problem to address at this time.

First of all, a difficult problem because it has many different ways of looking at it and independent problems that arise. Here I listed a few of them. You have a lot of irrelevant traffic that you have to filter out, we are not very much interested in that. In principle we're not only interested in how many hits we get but also how much time the clients spend on a page. Some particular browser-set things may cause problems to possible solutions of the problem. And then there is a huge problem related to proxies. Proxies are a weird thing of a live nature that cause metering problems. But what I will be talking about today, which is more relevant from a security perspective, is server-side falsification and client-side falsification. Of course web servers have an interest in declaring that they get many more hits than they really do.

The problem is important because a large part of Internet-related revenues today comes from advertisements, and of course before people invest money on advertising they want to know how much the site is worth. Therefore, obviously the sites need to show they actually have high access rates. So the commercial picture now is not very good from this point of view because only the very big, very well known sites get advertisement credit, like CNN, AltaVista, they get advertisements, people don't question whether they have high access rates or not, but there is a huge potential for medium-sized sites, for much more advertising that they get today. So this is an important problem today and there is very little research work, so that's why I became interested in the problem.

There are some commercial initiatives, Neilson, Netcount, Audinet is one I know better which is more recent. These commercial initiatives, as far as I know, are not very much into the security aspects of the problem. So they propose solutions but those solutions from my point of view are not secure. And there is some research work, this work presented at the WWW conference is not related to what I'm going to say today, but it does address some important issues such as the presence of proxies. And the works we generally relate to, which are also the only other works I found on the topic, are works by Naur and Pinkas, considering service-side falsification, and the work by Franklin and Malkhi that also tried to verify the client-side. Naur and Pinkas based their work on a K out-of N secret sharing scheme. They have an initial phase where they distribute the secret over the clients, and the clients send back to the server parts of the secret. So the server, in order to prove that they actually got so many hits, has to reconstruct the secret. So in order to prove that I got K hits, I must reconstruct the K out of N secret.

This is a nice idea from a research point of view but we believe it's not very practical because it requires the initialization of clients. And of course clients have no interest in somebody else's metering needs, and clients usually do not accept any extra time that is not related to their work.

Joan Feigenbaum: Do you think that otherwise Naur and Pinkas' scheme is implementable and sufficient. Do you think the secret sharing part of it, for example, is fine.

Reply: Actually one thing that puzzled me when I read the paper was that actually they get many clients so this K out-of N really becomes huge, so I don't know.

Joan Feigenbaum: That was what I was asking, is the arithmetic necessary to do this K out-of N for very large K and very large N prohibitive or are these servers doing things that are much more expensive anyway, I just don't know.

Reply: I don't know, I think that the other problem is more important, you'll never convince clients to do work for you.

Raphael Yahalom: But do both K and N have to be large or can K be arbitrarily small, even though N is large. I havn't read the paper.

Joan Feigenbaum: One hopes K will get large, right?

Stewart Lee: You want K as large as the alleged number of hits, don't you.

Reply: If you get one million hits, K will be one million. And I'm simplifying their paper, if you want a talk on their paper I need to prepare.

The work of Franklin and Malkhi is more interesting I think, and also it's very ambitious because it also wants to time the clients, so it wants to record how much time the clients' browsers spend on a page. And they do that with a special cryptographic technique known as a timing scheme. So actually what they do, when the client requires a page, they send the page but they add to the page an applet which will have to be executed by the client while the user looks at the page, and the precision of the answer will depend on how much time is spent on the page. And there is no way of computing a more precise answer without employing the computational resources that are related to the computation. So anyway the server will be able to tell how much time is spent by the client. And also another nice thing of this work is that it would be impossible to forge large quantities of clients' requests because that would be computationally very expensive. Of course the best part of the story is that you will force clients to devote all this CPU time just for this purpose which is not very practical.

Wembo Mao: I'm now in Cambridge, my web page still on in Bristol, and time is ticking, but I'm not reading it at all.

Virgil Gligor: If I click on your page and then just *xlock* and go home.

Simon Foley: Right, you click on a page, leave it there, you never know unless we go and check.

Carl Ellison: You could look for movement on the page, events.

Reply: You could do that. So that's a brief survey of the literature for you and I think it's a complete survey, if anybody knows of more work in this area please let me know. But that tells you that this is a very important problem, and there's very little work.

So here's my proposal which I believe it works but it's up to you to show me where possible problems are. I think that's the right way to do it.

So we have the server, we have a number of clients, that's the picture, and we also assume that our opponent cannot control a significant portion of the Internet. OK, maybe our opponent can control parts of it but the measure of those parts will be close to zero. So we are in the context that Professor Needham introduced this morning. We have a protocol and the parties who have to carry the protocol are actually those where the opponent hides. I mean we don't have an external opponent, the possible attack comes from the administrators of the website because they have the interest in making their hits higher than they really are. So it's like in electronic commerce but a slightly different scenario.

So the solution is based on this figure which, for printing reasons, is not on the paper for now but will be present in the final version, and this is how it looks. We attach to the server a tamper-proof box, and we can attach it directly, for example to the parallel port of the server. So that's the bad news. The good news is that everything else is as you see it. We have the Internet without any special requirements, we have clients that can be anywhere, this is a normal server which will have to do some extra things which I will explain shortly, and we have an audit agency that does all the checks, which is actually an HTTP server. Other good news is that the tamper proof box is very simple. It doesn't have to be a smartcard, because as you said there's no such thing as a tamper proof smartcard maybe. It doesn't have to be a smartcard, it can be a box of some size. It doesn't have to be very fast so even those differential fault-analysis things can be made much more difficult, and it has to be something very simple. What it has to do is the following.

The tamper-proof box receives from the web server one line of log, giving hit information. On receiving the line it generates a random bit, where the probability of one is very low. Then it outputs the bit back to the server and it authenticates the line (with a MAC for example). It marks the line if the bit was a one. So it gets one line, generates one bit. The reason which is behind this is the following. If there is a one, there will be a special check, if there is a zero there's no check. So it will be a probabilistic check. If we get a one we will ask the server to do a special thing, if there is a zero the server will behave normally, and will send back the http response to the client and everything will proceed normally. If the bit is a one the http server will have to do this: you must not serve the request, you must instead send a response which is technically an http redirect to the audit agency. So for those of you who don't know the details of the http standard, there are labels, headers, which allow http servers to force the client to reissue a request to another server and this is called an http redirect. So if the tamper proof box says, check this line, the server issues a redirect and therefore the client will re-issue the request to the audit agency, this time the audit agency will be able to check whether the request was real or not.

Stewart Lee: Doesn't that require secure software in the server.

Reply: No.

Stewart Lee: But you're asking the server to do something when the bit is a one. Why can't the server just change all ones to zero.

Reply: If the sever doesn't do what is required, we'll know about it in the end, because the tamper-proof box will authenticate the line to be marked.

So the audit agencies will sometimes, rarely, receive a log line, they will receive a redirected request. When they do receive a redirected request they store the log line, and then they redirect the request to the server again, because we don't want the poor client to lose the information. So even in those rare cases we will force the client to ask again for the page from the original server and finally get the information. So when this check occurs the client will have to do three http requests instead of one but this will happen only with very low probability.

Michael Roe: Are you sure that all standard-compliant browser implementations will be entirely happy about being given a circular loop of redirects from A to B and then back to A again. I mean this is the kind of thing that a good implementation might think, aha loop detection here, something severely erroneous.

Reply: Actually they don't, at least Netscape doesn't, I verified that.

Michael Roe: The other thing of course is that you can quite often legitimately get cases where the redirect doesn't get followed up on, either because the browser is broken, or the network died in between the redirect happening and the browser following it up.

Reply: Maybe, but if that were the case we could get the page from the server directly from the audit agency, maybe under a different protocol and serve the page direct.

Stewart Lee: When you send the redirected request back to the server, what is the probability of another "one" being generated?

Reply: OK, it's very low of course, so we just don't care about that. If we want to take care of your question directly we would just inhibit two consecutive ones.

Bruce Christianson: If you can't detect loops of length two, you can't detect loops of length four.

Stewart Lee: Indeed. I'm not worried about loops, I'm worried about this going on for a finite time.

Reply: So how does certification occur then? Periodically, for example after a month, we ask the server for the whole log file. They send us the log file, we also ask for the MACs of all the log lines. For each log line we have a MAC generated by the tamper-proof box, so we have log lines and MACs. Then for all log lines that are marked we check whether we had a redirect or not. Then if a high percentage of the marked lines did not generate a redirect we have a potential liar working with us, OK. So if many times, say 50% of the times, we have a marked line that did not generate a redirect, it means very probably a false hit which was generated. But it would depend on the policy of the audit agency, what to do in those cases.

A few more remarks. How do we generate the random bit? It could be a cryptographic random bit generation, such as those available, even just DES in output-feedback mode. Then we have to deal with the probability of one having to be low, actually some of you can suggest a specific random bit generation where the probability of one and zero is different, but it can be done. Actually, after talking with a friend I realised I think it doesn't necessarily have to be a cryptographic random bit generation, it can also be physically random bit generation, as long as it's unpredictable.

Joan Feigenbaum: But you don't want that, you want to control the probability of ones.

Reply: Yes, but we can control the probability of ones even on physical devices. So, I'm sure it works with the cryptographic bit generation, I'm not sure but it's possible the physical bit generation is simpler.

David Wheeler: It takes time.

Audience: At any point does the agency get any information from the tamper-proof box? Why can't the server just fake the logs and marked lines? Oh, sorry, well, you know, it's one of those freaky cases where no ones were generated and what are you going to do?

Reply: Well, of course, then you see we have a much lower rate of ones than you'd expect.

Michael Roe: The obvious thing to do is for the box to keep a chain of hashes of the log lines that were marked. And at the end there's a digital signature.

Clearly you ought to get something out of the box that then gets handed over to the audit agency at the end of the month or whatever.

Audience: When the audit agency does the whole computation and tries to see whether you're actually coming over the net or not, does it actually get any information out of the box. It gets the log files from the server, right, but does it get anything out of the box directly.

Reply: Well, they share a key, the audit agency and the box, because they have to check the MAC.

Audience: But does it get, for example, the number of ones that were generated.

Reply: If we have cryptographic random number generation, that's the answer to your question, because the audit agency will have the same key as the box for the generation of random numbers. If it's a physical random number generation then your question makes more sense. So, but maybe just by checking the number of ones we get we could, although maybe this worries me a bit, then I would I would go for the cryptographic random number generation.

Wembo Mao: The low probability of one, is it in order to limit the quantity of messages received by the auditor, the amount of traffic.

Reply: I just send the log lines with ones.

Dieter Gollman: No, the question was, the low frequency of ones, do you do it to limit the workload of the audit agency.

Reply: No, I generate a low number of ones because I want the check to occur only once in a while because it's bad for the clients, every time we generate a one the client has to stand a three way redirect, which is very bad. Imagine a site having all their clients redirect all the time, it would kill you.

David Wheeler: Can you reverse the process by doing as you said, holding the count in your tamper proof box, transmitting this when necessary, but only sampling the audit requirement. If you had a million you might sample a thousand, just enough to verify the right precaution was used. Are you with me? I'm suggesting you use the sampling process for checking, for auditing the count, in which case you don't even need to do it [all], if you've got one in a thousand originally then you might only have to check one in a hundred of all those samples.

Stewart Lee: A similar principle to statistical quality controls.

David Wheeler: The point is you can hold the count in your tamper-proof box, you can transmit this to whoever wants it, they have to check that the count is correct, you can do whatever process you were doing, but you need only do that on the set of the samples sent, and the statistical check will be enough.

Reply: Yes but that's what I'm doing. Isn't that a statistical check, I only check once in a while.

David Wheeler: Oh, yes, but I'm using it twice over and you're only using it once.

Roger Needham: His suggestion is that you having recorded one once in a thousand, you don't really do the check. And then do the check on another one in a thousand that were recorded.

Reply: Ah, OK, good point.

Stewart Lee: And it will be right 95 times out of 100 or something.

Reply: Well, I'm finished actually, but I can put out more problems just to make you more curious. There is one real problem in all this which I haven't mentioned, and I will, but just after taking a few more questions.

Wembo Mao: Then if the server wants the lists to be big it's quite easy to send http requests automatically so he can ask a person outside to send ...

Reply: Yes, that's an important point. If the website organisation has control of some IP addresses, from those IP addresses they can simulate requests and we can do nothing about it. I mean if Microsoft wanted to do this, they have computers everywhere, they could generate hits from those computers, you couldn't do anything about it. But they can only do it from IP addresses they really have, not spoofed IP addresses.

Audience: Actually you said that an opponent can control some parts of the network. Well if the opponent can control a lot of the part of the network that's logged on to the audit agency, they can do whatever they want.

Reply: Yes.

Joan Feigenbaum: That's one problem, but I think he's talking more about the phony hits problem, not about subverting the process, but actually generating more hits.

Audience: Yes, if I have access to the system box that goes to the audit agency then I can do anything.

Joan Feigenbaum: Yes, but that's a very extreme example.

Reply: I think that's not realistic.

Audience: I'm not so sure about that.

Reply: And the audit agency would do something back.

Audience: But you don't have to control the audit agency, you have to control the way to the audit agency, the way from everywhere.

Bruce Christianson: But this isn't a question of what is and what isn't, this is a question of what the particular threat this protocol guards against is. You think up another threat and you come up with your own protocol, yes? This one guards against the particular threat that was stated at the beginning of the talk.

Audience: Well the threat was that an opponent can control some part of the network, just not all of it. And I'm saying that yes, if it can control one part of ...

Bruce Christianson: The threat is, you're counting hits that weren't really there. The question about whether there are hits that really did happen but that you didn't actually want to count is a different question.

Michael Roe: Oh, no, not necessarily. If I can control the network I can fake hits. Suppose I'm in some minor country like Togo or something, and I can create a million users on the Togo Internet all of whom are frantically accessing these particular websites, providing genuine traffic for them. Unless you have some other auditing mechanism ...

Bruce Christianson: Then there's a protocol which says we don't count hits from Togo.

Michael Roe: But there needs to be this supporting stuff around this protocol to make it work.

Bruce Christianson: Sure, this is a brick, not a wall.

Reply: Anyway I think that's a good point. It's a good point but I think it's not realistic. If the attacker has complete control of all incoming packets to the audit agency, like if the attacker has a PC on the audit agency's line, then they can spoof IP, they can do everything.

Peter Landrock: In fact, how on earth would you know whether an access to the web server was a genuine one or a fake one. How would you determine that, what's the difference?

Reply: Well the idea here was that this approach seems to depend on the fact that you can't fake a large number of IP addresses. So when you force a client to connect to the agency then if it's a large number of IP addresses as opposed to a small number, you say well the server can't possibly have all those IP addresses and fake connections from all of them so must be true.

Simon Foley: If you rely on this, say for advertising, what if your competitor just turns round and sends in a stream of requests from faked IP addresses. So that what happens then is you end up with, everyone looks at your audit log and says, oh OK this is invalid. So where do we go back to, we have to throw the whole thing away.

Reply: Well it's a trust.

Simon Foley: It's not trust because I could come along ...

Reply: The audit agency represents, it's like with television polls, agencies say that NBC gets so much share, the agency has an image to defend.

Simon Foley: But I'm ABC, and I come along, and I send in a stream of requests to NBC from deliberately spoofed IP addresses. The mechanism eventually detects, or makes a log of the fact, that one of these was an invalid IP address, and then the problem is you have to throw away your entire log, you can't go back to the last valid one.

Bruce Christianson: Why do you have to throw away the whole log, why can't you take a proportion of valid bits.

Simon Foley: Because you don't know, how many did I produce, a thousand, ten thousand.

David Wheeler: Yes, but you can take the best estimate rather than throwing it all away.

Peter Landrock: I think now we've just reduced it to statistics. I don't know, I think it's very hot in here, I'm getting kind of thirsty.

Stewart Lee: What's the big problem?

Reply: There is a problem with proxies. If you have public proxies, those that accept everything, you can fool the system. But then the answer to that, after a bit of worrying, first of all the public proxies are very few and then you can also respond to the attack by looking at some special HTTP headers, which are I think X-Forwarded and X-Forwarded-By. The proxy usually forwards the request with that header filled in, so you should be able to detect that.

Delegating Trust
(Transcript of Discussion)

Bill Harbison

Centre for Communications Systems Research,
University of Cambridge

Delegating trust. We do use these words, we keep going back to them, you know, what is trust, what is delegation. I'm sure it will come to a great relief to many of you that I'm not going to talk too much about trust but I would like to point out that in common with many of the terms we use in security, there doesn't seem to be any general agreement on what it is. There's the orange book definition: a component of the system that can violate the security of the system. There are uses of trust that are indistinguishable from belief. There are attempts not to use the word at all. I have my own definition, but I won't bore you with that today. Needless to say though, this does cause confusion. It's almost unavoidable that we use the term but what I thought I would talk about today is the relationship between trust and delegation, because I think there is a relationship.

As a brief introduction I made a note of something that Joan said yesterday. She was talking about whether she would trust somebody to do something for her, and I believe in this context she was going to trust Steve Bellovin to sort out something for her. It merely occurred to me that when you talk about trusting somebody to do something for you in that way you are in fact delegating responsibility for that task to them. So I think there is a very strong relationship between trust and delegation and they are sometimes used, as Joan did yesterday, interchangeably. I think we do need to be cautious about how they're used, when they're used, and to try to get a little precision, We all start here - the dictionary definition. This just happens to be the one we had in the office: "gives duties, powers, etc. to another as a representative to depute or to authorise somebody as a representative." They are actually different things of course, and I think we will see that as we come on.

I thought, well perhaps we ought to say something about why it's important. I mean, why is delegation important? Well I think it's important because we can't avoid it. We're always delegating all the time. Sometimes knowingly, often unknowingly, we use the system to perform a task for us, you could view that as delegating the task to the system, and so I think it does bear some looking into what we're doing, how we're doing it, why we're doing it, and whether what we're doing is always the same thing.

Why do we delegate? Well, here are some of the reasons we might. Only we can perform the function for some reason, and therefore if for some reason we're not there, or we want to do something about it, delegate it. We delegate because someone else has a particular expertise or a particular facility that we don't have but which we need in order to complete the task we're doing. We may delegate formally in the sense that somebody may be capable of doing the task but the security policy of the organisation demands that we must take responsibility for it and therefore we must have some mechanism for transferring that responsibility to somebody else. And we often delegate for efficiency reasons. I mean this is the normal way that happens in many organisations, we delegate people to sign invoices up to certain levels, and so it's something we do do all the time. Are all those the same thing though?

We talk about delegating authority, we've already talked about delegating a task, I think it's quite clear it's not necessarily the same thing. If I delegate Bruno to change the tape, that's totally different from if I'm not here today and I delegate to Bruce the responsibility for running the whole workshop. I mean it's quite clear there are some differences in this and I hope that we can have some discussion about where they're essentially different, why they're essentially different, whether we ought to be using the word or not, and if we are, how do we distinguish between the two uses. And delegating a role is something that we do see all the time, again in normal organisations managers go on holiday and they often don't just delegate tasks, they will delegate their function as Head of Department, say, to somebody because the function of Head of Department has some meaning in the organisation, it has some process involved with it.

Joan Feigenbaum: When and why would you delegate responsibility without delegating one of those other three things? So you want somebody to have responsibility without power?

Carl Ellison: It happens all the time in corporations.

Joan Feigenbaum: I know, that's what happens to me and I wondering why.

Carl Ellison: You might also call it delegating liability or delegating blame.

Reply: So what is it? We're transferring or assigning responsibility for part of our function. Wait a minute, that sounds like the same thing doesn't it? But is it? Is transferring and assigning the same task? Maybe this comes close to what you're asking there Joan, because in one there is an implication that you're not just transferring control but also the responsibility, and in the other you're not.

Joan Feigenbaum: Would you ever delegate responsibility without control.

Reply: Oh, they do it in government all the time.

Joan Feigenbaum: I know they're doing it in government all the time, but it may be one thing we can clarify for all these people, why you should never need to do that. We'd increase security.

Bruce Christianson: Well you might want for example to have a sub-contractor's relationship where the sub-contractor can prove that they complied with ISO9000 and the prime contractor could not have interfered with it because of the nature of the quality assurance process but the prime contractor retains responsibility for the performance of the subsystem.

Reply: Responsibility is the word that we haven't brought up yet and it is a very interesting one.

Joan Feigenbaum: Actually insurance companies, in some sense, take on responsibility for that control.

Bruce Christianson: But this is a situation I'm describing where the prime contractor will resist bitterly any attempt to give them control over the subcontractor. They're happy with the responsibility because they're getting paid for it. They want to be able to prove in court that they could not have subverted the quality assurance procedures.

Reply: And of course there are many cases where this is hidden from you, there are cases where you think you're dealing with principal A, but principal A whose company X has actually delegated responsibility for breakdown insurance which is notionally in its name to an insurance company and if you have a claim you find yourself talking to the insurance company rather than the manufacturer in the end. And so you very frequently do get transfer of responsibility without control by the person transferring the responsibility.

Michael Roe: There's a separate issue of liability creeping in here and certainly with an insurance company you are delegating the liability without delegating any control at all to them. But that may not be the same as responsibility.

Reply: I wrote down a question to myself yesterday. I think I know what the answer is, but let me just read you what I said: "Are these two statements equivalent. (1) The President authorised the action. (2) The action was authorised by a key assigned to the President."

Carl Ellison: They better be different.

Reply: They are different I would maintain but you see when we talk about delegating particularly in the security arena we think that they're the same thing, that when we delegate the key that's actually delegating responsibility.

Carl Ellison: There are even some laws that try to make them the same.

Reply: Do they succeed?

Carl Ellison: They haven't been tested in court yet but the Utah digital signature laws are trying to make those the same. And that's something against which some of us are trying to fight very hard because they're clearly not the same.

Reply: I maintain that when you put it in such blank terms as I've just read out it seems quite clear that they can't be the same thing, because you can see already that there might be an intermediary involved in the second where there wasn't in the first, and again, when we talk about delegation often there are intermediaries in the system. As I said if we use a system to perform a task on our behalf we are delegating responsibility in some sense to the system but who is accountable in the end. Are we accountable for a transaction which the system screws up because we delegated it to it?

David Wheeler: I think you're blurring a distinction, which is that personal delegation is quite different to computer delegation. We pretend they're the same but in fact they're different. In most cases of computer delegation you could almost certainly use the word "use" instead of "delegation". That doesn't happen in the case of personal delegation. The rules seem to be different but we're trying to pretend they're the same.

Reply: Well, I would maintain there are some circumstances where you don't know whether a person's actually done the task or whether a computer's done the task at the outcome. So there's a question of whether the outcome is important or whether the intermediary is important.

David Wheeler: No, I was using the word delegation in this context: you can do things personally. A President authorising somebody is quite different from a computer authorising someone.

Reply: But supposing I authorise the computer?

David Wheeler: That they're the same, I think, is possibly wrong. To try to achieve the same results might possibly be right.

Carl Ellison: I'd be happy to say that you use a computer to perform some function like creating a digital signature. What makes me uncomfortable is when people make the assumption that if you are using Carl Ellison's computer to do this then Carl Ellison must have been the person who used that computer. The assumption in the Utah law is that a key certified to me is something for which I am responsible no matter who used it. They don't accept the concept of repudiation of ownership of the private key and so they don't permit repudiation of signatures.

Reply: I'm just going over the events when someone steals your computer.

Carl Ellison: Oh, if someone physically steals your computer them presumably you should go revoke your certificate.

Reply: What if they steal your computer and do it before you know it's missing?

Carl Ellison: But if they walk into your room and use your computer and then leave before you get back to your room then you have no defence against that.

Raphael Yahalom: No, no, it's part of the evidence. Come on it's up to the discretion of the court to consider the whole picture, [part of it] being that your own personal keys were used. I mean I'm not a lawyer but that's my interpretation.

Bruce Christianson: But it's the question of presumption, the presumption is you signed it.

Raphael Yahalom: If you show physical evidence that your computer was broken into surely nobody would find you responsible.

Simon Foley: You have a duty of care when making the service available. If you have to go to court, but you can show that you put sufficient controls in, you may be OK. But if you haven't made that duty of care and you make the service available and [the security is] easily bypassed then you have a responsibility.

Joan Feigenbaum: The opponents of the Utah law I think are mostly uncomfortable with the fact that this accepted duty of care standard is not in place, but the law making you responsible is going on the books. I think it would be better to have a law that said you are responsible unless your key was stolen, or unless you signed under duress, or something like that. But that's not what the law says, the law just says you are presumed to have signed it.

Raphael Yahalom: There is an assumption that there is going to be an interpretation by the judge or by the jury who would look at the whole picture. It's like computer break-in, there are similar issues: you have the law and then you have the interpretation, which is where a lot of the action is.

Larry Paulson: Is it similar to the treatment of these signature stamps, these physical stamps which are used to sign cheques. I worked for a company where they kept this thing in the safe and if you stamped it on a cheque it counted as the boss' signature, so this is morally the same as a digital signature and if it got stolen it was pretty bad.

Carl Ellison: Exactly, and there are cheque embossing machines which are also kept in safes, but the private key on this laptop is not being kept in the safe.

Bruce Christianson: Well that's an incredible statement of fact.

Carl Ellison: And furthermore the software on this laptop that is capable of using that private key if I ever provide the pass phrase that opens up the private key, is not under my control. It happens to be under Microsoft's control, on this particular laptop. It could have been infiltrated by viruses. All of these things are outside of my control and the digital signature law does not recognise those possibilities. Instead it says if I accept the certificate for this key of mine then I accept the responsibility, the liability for everything that that private key ever signs until I revoke the certificate.

Reply: I think we've got a couple of developments. Delegation implies possession or ownership of something that we can assign to another party. If we take the traditional definition it also implies a specific conscious act on our part to do something as well, and perhaps that's what's missing from this law. The implication in delegation is the standard thing that there is a conscious action that we're doing. Notice it doesn't actually say anything about the person receiving the benefit of that action. Again those of us who've worked in large organisations know that frequently the first time that you know that you've been delegated something is when your copy of the memo that went out to everybody arrives on your desk.

Bruce Christianson: That's because you've delegated your power to accept responsibility.

Bruno Crispo: This may be because in the definition it's implicit that the grantee knows he gets some power.

Reply: Well, as I say, I just noticed that it's absent. Perhaps we do need to make it explicit in protocols that when we are delegating something there does need to be activity on behalf of the delegatee.

Virgil Gligor: Maybe occasionally when we do delegation we don't necessarily want the delegatee to accept it. For example, the model of log-in as a delegation to your PC, to the system, to operate as you. You don't necessarily want acknowledgement from the PC back that it accepts the delegation or refuses it. That's because we are choosing to model the log-in by delegation. When is acceptance artificial and introduced by us because that's how we chose to model things and when is it real? Does it always have to be there?

Reply: The only way I can answer that question is to say that I suspect that it depends on how you are considering accountability. Whether you want plausible deniability in the sense of "I'm sorry the computer screwed up, it all went wrong, not my fault, honest guv, you know, it was the computer that sent you this million dollar bill for your gas last month."

Roger Needham: I would like to pick up on David's point, machines and people not being the same. In all of this sort of discussion, ever since I think Schroeder and Salzer in 1975, the term "principal" has been used for the thing which is authenticated, the think which takes part in these kinds of protocols we're talking about, and they invented it, or adopted it, I'm not sure which, because they realised there was a lot in common if you're talking about these sort of protocols between people and computers and programs and things like that, and it was convenient to have the notion of the principal for that which takes part in things like authentication protocols. It may be that that's a useful abstraction until you come to talk about delegation when it stops being useful at all.

Reply: Well, I take that point but do you not think that the whole advent of digital signatures might perhaps blur this distinction again. This seems to be what's happening in certain places.

Virgil Gligor: I know of a lot of analogies, for example, protection models and mechanisms can be modelled as real estate law. We can make a lot of analogies and a lot of them are wrong, and this is I think what David is pointing out and Roger and I are picking up on it. I think that there may be some limits to these analogies and there are some limits in modelling reality. In other words we have to recognise when it's the model of reality and when it's real and maybe this is a good place to make that distinction.

Reply: You may indeed be right but there are states, there are countries passing laws about the treatment of digital signatures and their use.

Virgil Gligor: Here is the difference: the difference is liability. There is liability in law. By the way this is real, this is not a model. The way liabilities are assigned is to the party that can do something about the effects of the error, in other words you never assign liability to somebody who had no control over the outcome. So, for example, if I have no control over what happens to my signature, to my key, liability law says I am not liable for that particular piece of software that's used my key, so in some sense, Rafi is right, that this notion in the Utah law of being solely responsible for it, is never going to be literally interpreted because it's interpreted in the context of other laws that regulate who is responsible and who is liable. So it may be that we should look a little bit at delegation from that point of view, from the point of view of trust liability for the actions.

Raphael Yahalom: I want to pick up on Virgil's and David Wheeler's points. I perfectly agree, I think once we talk about delegation, we move more to the interface between the technological and legal domain. Delegation makes sense in a legal context, in which case the principals of interest here are what I would call legal entities, and legal entities would be people or organisations. You would never sign a contract with computer hardware or software. I think one test to determine what is the relevant principal here is to ask who would be able to sign a contract. The whole issue of delegation should be thought about in the context of *a priori* contracts which have been signed explicitly or implicitly.

Stewart Lee: One of the more significant principles of security is that everything must have some person behind it who is responsible for it. Without having a person responsible for a thing then there's not a lot of redress.

Raphael Yahalom: A person or organisation?

Stewart Lee: A legal person. Consequently it is the concept of delegating some real life, outside the computer, responsibility to a program needs to be looked at. Because then can you say who is responsible?

Reply: Well there's putative delegation going on and it comes back to who's accountable in the end.

Stewart Lee: When you put up your first slide, you said your title was delegating trust and I think it should be trusting delegation.

Virgil Gligor: Suppose you that you are investing in a piece of software to solve a system of partial differential equations, and you get back the result, is that trusting? You can verify that the result is correct simply by plugging the solution into the equations, which is much easier that solving it. Is trust like faith that you cannot verify, or is it something that you can verify totally or partially?

Reply: My definition of trust is effectively a measure of what you don't know and can't verify. Trust is a phrase for something that we cannot effect or it's not economic for us to do so.

Stewart Lee: It's more complicated than that, trust is an extrapolation of what we know into what we don't know, based upon experience.

Bruce Christianson: Signature is another good example where we're aware of the fact that there's a difference between our normal social life and what goes on in the electronic world but there's actually two separate sets of differences. One is that the mechanisms that we use for things like signatures and handing off responsibility are different in the two cases. We use different technologies and the technology has different second-order effects that cause ripples that spread outwards. But the second element of it, that we don't always look at or we get confused about, is that the problem we're trying to solve is subtly different in the two worlds as well. In other words, what a signature ought to be in the electronic world is different from what it ought to be in the world of signing documents and passing them round, and we tend to obsess on the fact that the technology forces certain different mechanisms on us.

Reply: It's not just that, it's that some of the legal people are trying to make the two things equivalent and I think David's quite right, what we're pointing out is that you cannot, to some extent, make these things equivalent. The whole of the rush towards digital signatures, electronic trading on the Internet is trying to make the two effectively equivalent and I think what you're saying, and Roger and David are saying, is they cannot be equivalent.

Stewart Lee: Sure, the computer industry is full of words that have been imported from real life, and that have a different meaning. The same word is being used for a different meaning. Another one that is obviously related to this is watermark, which has a very different meaning in the digital world than in real life.

Reply: I don't disagree with that, all I am saying is that in the non-technical sphere people are trying to make this equivalence and I think that is what is causing a lot of the problem, because we are talking about saying, because we call an apple and an orange a fruit they're the same thing.

Dieter Gollman: I think it's three discussions. The discussion in the world of corporate culture, that's where we started off: politics and office politics. You have the world internal to a computer system and you have the interface between these two worlds. That's where, for example, digital signatures come in. I don't see a lot of attempts at keeping these three worlds intellectually apart in our discussions. We are jumping in and out, and back and forth, and contributing to the confusion.

Reply: But this is one of the points I'm trying to make. Because we speak language and we speak English and we speak a term, we think we know what it means and we use it in different contexts and think it means the same thing in different contexts. But it doesn't.

Dieter Gollman: You know my theory, once you use a word in the English language, you don't know what it means.

Bruce Christianson: The really dangerous thing is that we think that if we could come to a consensus about what it meant, this would help us in some way. That's the thing that we somehow have to get beyond.

Virgil Gligor: How about trying to do a taxonomy of delegation to find out how many forms do we have, and which are real and which are imaginary in computer systems.

Reply: Yes, I agree, I think that it needs to be done and looked at in computer systems because as you go through a computer system you see putative delegation going on all over the place in the general sense of the word.

Virgil Gligor: Three areas: in real life, in computer systems and in models of computer systems. Occasionally models of computer systems try to come up with a single primitive concept that explains everything under the sun. That's considered to be a good thing. So, if you look at the Lampson, Abadi, Burrows and Wobber paper, they tend to do that, they tend to model different aspects of cryptography using as few primitives as possible. There is obviously something missing in that approach, but it is not clear what is missing, at least in the area of delegation, unless we do a taxonomy in computer systems.

Carl Ellison: The idea of doing that taxonomy is very appealing to me but I am reminded of an experience we had in the SPKI working group. About a year ago, we addressed the issue of delegating rights versus responsibilities, and realised that for many things there were both rights and responsibilities involved in any delegation. We also realised that if you're delegating, let me not use the word responsibility, because of the definition today, if you're delegating liability in a certificate then the signer of that certificate should really be the subject rather than the issuer. It's only when you're delegating rights that you want it signed by the issuer. So for a while we were proposing that all certificates should be signed by both the issuer and the subject so that any responsibility was accepted and any rights were properly granted. That proposal died an ugly death because people considered it first of all too complex and secondly a concession to lawyers and there was a strong belief that we should design for the technical community not for the lawyers. I'm not sure that's a correct decision, but that was the thinking that went behind our dropping the proposal to have certificates signed by subjects.

Joan Feigenbaum: Why is everybody assuming that one has a choice about accepting responsibility? It's not always true in the sense that, for example, at the time you get a certificate that delegates some responsibility to you, perhaps also some rights to you, you may really have no choice because of some previous contract or relationship that you entered into. You might be required at this point to accept whatever responsibility this delegator is delegating to you.

Bruce Christianson: Sure, policy may require you to sign but the analogy is that if a file store is given a request for a particular page you might say well it would be a good idea if the file store signed the page that it sends back to certify this really is the right page and is given in response to your request. Now that's not saying that the file store has a choice about whether it gives you the page back or not, you've got a client-server relationship with it, if you ask nicely it has to give you the page. We're arguing about whether the audit trail should show some evidence that you could appeal to, to show that it really was the file-server and it really was a response to that request.

Joan Feigenbaum: OK, so if there was some more efficient way than having the delegatee sign, that would be OK as well, you're not implying by saying that the delegatee has to sign that he actually has a choice about whether to accept this delegation.

Reply: No. I have shifted around because I think that is what we tend to do. There's the Morrie Gasser paper on delegation. When you read these papers and you try to take the concept and follow it through the logic it subtly changes its meaning as it's applied through the different examples. All I'm saying is that is very dangerous. In the same way that we use trust, and perhaps we should stop, we should find some other word to use, maybe accountability, maybe transfer, maybe something which has less subjectivity about it which we can say when we do this, this is precisely what we mean, and this is precisely what happens. Because we are still talking about people using systems to perform some task, we need to quite specifically tackle the problem of accountability, i.e. who will be accountable under what circumstances for the behaviour of the system?

Bruce Christianson: Part of the problem with the delegation shell game is that we don't know how many peas there were to begin with.

Reply: Well, I almost put an example up here again to ask people what they think this means in terms of delegation. Principal A delegates authority to some delegate A, principal B delegates authority to some delegate B and the two delegates go off and interact in some way, maybe with a third party. Now this does happen in real systems, certainly in client-server systems.

Bruce Christianson: There's also the problem where a red manager and a blue manager attempt to delegate a key to the same secretary. Now the secretary has got to have the power to refuse the second delegation even if the policy says she should take it.

Reply: Because of higher policy issues.

Bruce Christianson: Everybody understands this in the physical world even if they've got a physical piece of plastic with the electronic key on it but as soon as its noughts and ones written on a piece of paper common sense goes out the window.

Bill Harbison: So, what does delegating trust mean? Transferring the responsibilities for our uncertainties to others? So who is accountable?

Delegation of Responsibilities
(Position Paper)

B. Crispo[1,2]

[1] University of Cambridge
Computer Laboratory
England
[2] Department of Computer Science
University of Turin - Italy

1 Delegation of Responsibility

Delegation is usually seen [1,5,2,8,10] as delegation of rights, where the term *delegation* is used to refer the process whereby a principal authorises an agent to act on her behalf, by transferring a set of rights to the agent for a specific period of time.
Here we would like to propose an other form of delegation: *delegation of responsibility* with the following definition that is slightly different from the previous one: *delegation* is used to refer the process whereby a principal authorises an agent to act on her behalf, by transferring a set of rights to the agent for a specific period of time - during which the principal can no longer exercise these rights.

Delegation of rights assumes that a principal, A, has a set of rights G and she delegates a subset of them, G', to an other principal B who can act on behalf of A to exercise that particular set of rights G'. A is always responsible for G', but with delegation she shares these responsibilities with B.

We use the following terminology: the principal that delegates is said to be the *grantor*. The principal that is authorised by the grantor to use some rights is said to be the *grantee*. The target that accepts the last operation invocation of a cascaded operations chain is called the *end-point*.

To whom a principle is responsible for her rights depend on the security policy. Usually the grantee is responsible to the grantor about the rights he got from her, but there could be cases where the grantee is responsible to a third party or many other more complex agreements.

There are a lot of situations where this sharing of responsibilities, that introduces uncertainty for auditing purposes, is not desired.

We use the term responsibility to mean something different from what the term is most commonly used for. We will not determine who is going to pay a bill, but rather to make sure that the bill will be delivered to the person accountable for it. We cannot technically establish who is going to be legally, morally or socially liable for an action. Our goal is, more modestly but still quite difficult, to determine who, technically, performed a specific action.

Delegation of responsibility takes place everytime a principal has to rely upon some other party to complete a transaction. She relies upon an other principal but she does not trust him.

Following we describe a couple of examples where this new form of delegation is needed.

Let suppose to have a service provider that offers a set of services to her customers. The customers demand for a new service. The service provider finds more convenient to delegate this service to an other party rather than to provide it by herself. Customers do not want to see this delegation, i.e. the new service has to have the usual interface of all the other services. The responsible for all the services for the customers it is always the service provider (grantor). Anyway the service provider wants to be able to distinguish if a disruption of the new service is its responsibility or the other party (grantee) one.

In order to do this we propose a new form of delegation: *delegation of responsibility*. What it is delegated are not only the rights but the right with the attached responsibilities.

An other practical example how delegation of responsibilities differs from delegation of rights is the following.

Let us suppose the case of a bank manager that possesses the physical key to open the vault. The manager needs to go away for a week and she delegates her chief cashier to open the vault in her absence giving to the secretary a copy of her key. This is represented in security literature as delegation of rights. But the delegation could be implemented also in another way where the manager gives to her secretary the original, not duplicable, physical key to her, to delegate to her the power to open the vault in the manager's absence. This second case is what we define as delegation of responsibility.

The difference become even clearer if the vault is mysteriously emptied out while the manager is away. In the former case both, manager and secretary, will be put in jail (or none of them if the principle of doubt held). In the latter case, the manager may be fired because she did not choose carefully enough her secretary but it is the secretary that will be put in jail.

This is a typical example of delegation of responsibilities and not of rights. In the banking practice there are a lot of examples where for security and auditing purposes more than one independent parties cannot execute themselves a transaction but their authorisation is necessary for the execution of the transaction by one or more responsible party.

We are not saying that delegation of responsibility must substitute the concept of delegation of rights, which is useful and appropriate for many situations, but rather we pointed out that both meanings of delegations are necessary to provides mechanisms to solve a wider variety of problems.

2 Critics to the Existing Works

In this section we will show what are the assumptions that need to be redefined in some of the existing works on this subject, [1] and [5], in order to be able to represent delegation of responsibility.
We will refer mainly to those two papers because, for what we know, they are the most complete examples of theoretical framework about delegation. In security literature there are relatively few papers investigating delegation issues, and most of them [2,4,6,9] focus only on mechanisms to implement delegation without critically consider the concepts underlaying delegation.

2.1 Relation *"speaks for"*

Relatively to [5], our first observation is concerned with the definition of the relation 'speaks for' and the statement 'says' given in that paper:

> There is a fundamental relation between principals that we call the 'speaks for' relation. A speaks for B if the fact that principal A says something means we can believe that principal B says the same things.
> ....
> We use 'speaks for' to formalise indirection; since any problem in computing can be solved by adding another level of indirection..

This definition of 'speaks for' is too strong. Implicit in this definition is the assumption that 'speak for' contain also the 'says' concept in it. If A speaks for B than B says what A says.

We would like to emphasise that this could be true but it is not necessarily true and with this definition of 'speaks for' it is impossible to differentiate the two cases.

This problem is explained even better looking at the formal definition of the relation 'speak for':

$$(\mathrm{A} \Rightarrow \mathrm{B}) \equiv (\mathrm{A} = \mathrm{A} \wedge \mathrm{B})$$

and from that

$$\mathrm{A}\ \textbf{says}\ \mathrm{S} = (\mathrm{A} \wedge \mathrm{B})\ \textbf{says}\ \mathrm{S} = (\mathrm{A}\ \textbf{says}\ \mathrm{S}) \wedge (\mathrm{B}\ \textbf{says}\ \mathrm{S})$$

Thus 'speaks for' implies that all what B says is said also by A. This is true when only rights are delegated but it is not when responsibility is delegated.

2.2 Operator *"for"*

In [5] delegation is expressed with the operator *for*, that it is axiomised as following:

$$A \wedge B \mid A \Rightarrow (B\textbf{for}A) \tag{1}$$

$$\textbf{for} \quad \textit{is monotonic and distributed over} \quad \wedge \tag{2}$$

$$A\textbf{for}(B\textbf{for}C) \Rightarrow (A\textbf{for}B)\textbf{for}C \tag{3}$$

$$(A\textbf{for}B)\textbf{as}R = A\textbf{for}(B\textbf{as}R) \tag{4}$$

A to delegate to B it makes (5) A **says** B | A ⇒ B **for** A. Then B must accept the delegation by making (6) B | A **says** B | A ⇒ (B **for** A). This explicit action by B is needed because when (B **for** A) says something, the intended meaning is that both A and B contribute, and hence both must consent.

Given (5) and (6), B can speak for (B **for** A) by quoting A.

$$(\mathrm{A} \wedge \mathrm{B} \mid \mathrm{A})\ \textbf{says}\ \mathrm{B} \mid \mathrm{A} \Rightarrow (\mathrm{B}\ \textbf{for}\ \mathrm{A})$$
$$\mathrm{B} \mid \mathrm{A} \Rightarrow (\mathrm{B}\ \textbf{for}\ \mathrm{A})$$

Those axioms are too strong to represent delegation of responsibility. A and B not only do not need to both contribute to a statement but they have not to. Or more precisely their contribute must be well defined and not shared between them.

Delegation of rights allows a principal to delegate some of all her authority to an other principal. In order to do that the grantor requires a level of trust in the grantee while in delegation of responsibility this trust not only is not required but in many cases grantor and grantee have conflicting interests on the rights exercised by the grantee and at the same time both of them are interested on the fact that delegation takes place.

Thus in delegation of responsibility the grantor hands off the rights with all the responsibilities attached to them to the grantee. The grantor has simply to rely upon the grantee but not to trust her. This is a crucial difference because reliance can be monitored and verified while trust cannot. A trust relationship can be only interrupted (by revocation) but during its validity cannot be verified by a third party. This is an essential property for non-repudiation for example.

Revocation may reflect also those differences. With delegation of rights it is reasonable to assume that the grantor exercises the right to revoke a delegation token, previously issued, because she retains responsibilities. With delegation of responsibility because all the responsibilities are handed off also the responsibility to revoke a delegation token can be delegated by a third party different from grantor and grantee. This suits very well the principle of separation of duties that should be always enforced where possible in security system design [3].

We note a divergence between delegation of rights and how delegation is performed in real world. In most of the practical case, and almost always in business practice, when a grantor delegates some of her rights to a grantee, then the grantee to exercise those rights is usually requested to sign with her handwritten signature. Thus even if responsibilities are often shared between grantor and grantee the real world seems to employ always mechanisms to allow to recognise if the grantor or the grantee exercise a delegated right.

On the other hand the delegation of rights, as it described in almost all the security literature does not seem to allow this distinction. The delegation key, key used to exercise the delegated right, is known by both grantor and grantee, leaving room to undetectable misbehaviour, also in case that proper audit trail are maintained.

An example of delegation of rights in the real world practice is the case where a manager gives a copy of his office's key to his secretary, delegating her the right to enter in his office in his absence. Here, as in the electronic case, there may be situations where it cannot be distinguished who, the manager or the secretary, got into the office.

We have also to mention the containment problem. Mechanisms that electronically implement delegation of rights in order to prevent the grantee to delegate, to some unauthorised party, the rights she got, usually require that the grantor specifies the name of the grantee in the delegation token. The end-point will accept the delegation token only if presented by the principal specified in the relative token.

2.3 Differences between Public-Key and Symmetric-Key Cryptosystems

The logic presented in [1] and [5] does not seems to capture the differences between symmetric-key and public-key cryptosystems with respect of delegation. Let us suppose A and B two principals and S a trusted server. A and B share a key with S, respectively K_{AS} and K_{BS}.

According with that logic we can write that: $\{K_{AB}, A\}_{K_{BS}}$ means that K_{BS} **says** $K_{AB} \Rightarrow A$. With symmetric key, K_{AB} is a secret key, then 'speaks for' and 'says' coincide. Using the same notation for public-key can lead to some subtle mistakes. If we write $\{K_{A^+}, A\}_{K_{CA^-}}$ as equivalent of K_{CA^-} **says** $K_{A^+} \Rightarrow$ A, we have to remember that is K_{A^-} that 'says' and not K_{A^+} Thus the certification authority CA says that K_{A^+} 'speak for' A while actually A 'says' with a different key: K_{A^-}, unknown to the CA.

2.4 Passive Principals and Active Principals

Finally the logic in [1] does not catch the differences, between delegating a passive entity (e.g. a key) or an active one (e.g a principal).

From the definition of 'speaks for', writing B $\Rightarrow$ A, where A and B are principal, means that whatever B says implies that A says the same. This requirement is difficult to be satisfied, because requires A to completely trusts B, in the sense that she trusts the fact that B will say only things she will say, or B must have restricted capabilities, in particular he can say only things that A will say. This is very difficult to enforce if A and B are active principals.

While if we write $K_{AS} \Rightarrow A$, where K_{AS} is A's secret key shared with the grantor S, than it is much more natural to assume that the key that speaks for A will say only thing that A will say.

2.5 Roles

Some papers [5,2] introduces *roles*, as mechanism used by a grantor to limit her authority that she delagates to the grantee. Using the notation in [1], A **as** R, meaning the principal A acting as the role R, is a weaker principal than A because he has only a subset of the whole rights A has.

So writing B $\Rightarrow$ (A **as** R), it is a weaker statement than B $\Rightarrow$ A, because it means that whatever B says implies that A in the role R says the same, but B cannot say things that A can say acting outside the role R.

Roles do not solve all the problems. In fact, where principals and not passive object are involved, there could be situations where two different principals A_1 and A_2 with different security policies (e.g. A_1 must always terminate operations before A_2 starts hers), delegate two different roles to the same principal B: B **for** (A_1 **as** R_1) and B **for** (A_2 **as** R_2). It is easy to see how B can break the grantors' security policy or how difficult is for the grantors to enforce their security policy. This delegation mechanism could lead to an even worse situation in case A_1, A_2 and B have conflicting interests. It is also not easy to detect when situations like this may happen, because there could be chains of delegations.

Let us suppose to have the following two independent chains of delegations:

$$\text{B } \textbf{for} \text{ A } \textbf{as} \text{ R and C } \textbf{for} \text{ B } \textbf{as} \ R_1$$
$$\text{C } \textbf{for} \ A_1 \ \textbf{as} \ R_2$$

The independent grantors A and A_1 cannot determine or detect if the delegated rights of C acting as R_2 are in somehow influenced by the delegated rights of C acting as R_1. Sometimes just the knowledge of the existence of those rights could be enough to leak confidential information. Moreover C himself may take advantage about this situation and surely he is not strongly motivated to report this anomaly situation.

A cross checking of all the roles delegated by a grantor is not only an expensive procedure but sometimes it may not be possible because it may require access to restricted information.

We wish also to point out that also the validity period of the delegation token is worth to be discussed more carefully. The usual assumption is that the lifespan of the token is shorter than the grantor's life. There are a lot of applications [7] where this is not the case.

3 Conclusion

We wish to remark the importance of discussing novel definitions of delegation and its importance in designing security systems that may offer services as non-repudiation.

We are actually working to redefine the definition of the logic that we criticised here in order to represent delegation of responsibilities. We are also implementing this concept using different cryptographic mechanisms. Our suspect is that there is not a particular cryptosystem that can offer solutions or features that other cryptosystem cannot provide. It is rather question of system constraints and specific application requirements which cryptosystem to use.

Acknowledgement: The author would like to thank Microsoft Research Ltd. for supporting his research at the University of Cambridge. He would like also to thank: Bruce Christianson, Roger Needham and William Harbison that were at the origin of many ideas discussed here. Simon Fowley, Francesco Bergadano and Geraint Price for the useful discussions and suggestions to improve the presentation that follows this paper.

References

1. M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A Calculus for Access Control in Distibuted Systems. *ACM Transaction on Programming Languages and Systems*, (15):706–734, September 1993.
2. B.C.Neuman. Proxy-Based Authorization and Accounting for Distributed System. In *Proceedings of the 13th International Conference on Distributed Systems*, May 1993.
3. B. Crispo and M. Lomas. A Certification Scheme for Electronic Commerce. In *Security Protocol Workshop*, volume LNCS series vol. 1189. Springer-Verlag, 1997.
4. Y. Ding, P. Horster, and H. Petersen. A New Approach for Delegation Using Hierarchical Delegation Token. In *Proceedings of the 2nd Conference on Computer and Communications Security*. Chapman and Hall, 1996.
5. B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in Distributed System: Theory and Practice. *ACM Transaction on Computer Systems*, (10):265–310, November 1992.
6. M.R. Low and B. Christianson. Self Authenticating Proxies. *IEE Electronics Letters*, 30(2):124–125, January 1994.
7. C.P. Martin and K.Ramamritham. Delegation: Efficiently Rewriting History. In *Proceedings of the 14th International Conference on Data Engineering*, pages 266–275, April 1997.
8. M.Mambo, K.Usuda, and E. Okamoto. Proxy Signature for Delegating Signing Operation. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, 1996.
9. M.Mambo, K. Usuda, and E. Okamoto. Proxy Signatures: Delegation of the Power to Sign Messages. *IEICE Transaction on Fundamentals*, E79-A, September, 1996.
10. V.Varadharajan, P. Allen, and S. Black. An Analysis of the Proxy Problem in Distributed System. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1991.

Delegation of Responsibility
(Transcript of Discussion)

Bruno Crispo

University of Cambridge

Today I will present for the first time a work that is still under development. I will talk about some issues related to delegation.

Yesterday Carl Ellison said that people usually talk about trust but they never define it. So I decided to start the day with a definition.

During the last few months I have studied many papers investigating delegation in the security area and all of them use delegation generically as synonymous with a particular form of delegation, that is delegation of rights. Delegation is then defined as the process whereby a principal authorises an agent to act on her behalf by transferring a set of rights to the agent for a specific period of time. We defined the grantor as the person who hands over the rights, the grantee, who accepts them, and usually what is handed over is a subset of rights owned by the grantor to the grantee while the responsibilities for these rights are always shared between the grantor and the grantee. There is always a share of responsibility on the rights and usually both of them, grantee and grantor, have the capability to exercise the delegated rights.

We want to point out that there are a lot of situations where we do not desire this uncertainty due to the share of responsibilities. What we want is a mechanism able to detect who is the principal accountable for a certain action.

Another possible definition of delegation, that does not substitute the above one but I think it may be equally useful, is delegation of responsibilities. Delegation of responsibility is defined as the process whereby a principal authorises a principal to act on her behalf, for a specific period of time, during which the principal can no longer exercise these rights. The fact that the grantor cannot exercise these rights is the crucial point because this is a necessary condition for the grantor to hand over not only the rights but the responsibility attached to the rights.

The principles underlying these two different type of delegations are very different, because delegation of responsibility seems to fall in those cases where a principal has to rely on a third party to complete her transaction, but he does not have to trust the other party, while with delegation of rights, trust is involved. The grantor must carefully choose the grantee because once she has handed over the rights to him, it is very difficult if it happen, to distinguish who, between her or her grantee, has abused these rights. It is a trust relationship, that may be revoked but hardly monitored or audited.

There exist many examples of this kind of delegation, in real life.

As I say I have studied some works and papers on delegation and a lot of them focus their attention on the implementation issues but very few papers investigate the definition of delegation. As a result there are a lot of papers that just implement in different ways delegation only of rights. There is no paper that has a different point of view on delegation. One of the few frameworks investigating the formalisation of delegation is the work done by Abadi, Lampson et al., at DEC, at the beginning of the nineties. They specify a logic and a calculus to represent several issues related to distributed systems, among these issues also delegation. However the calculus proposed is able to represent just delegation of rights.

I will give now a brief overview of what are the issues of this logic that need to be re-discussed in order to be able to represent also delegation of responsibility.

One of the basic blocks of their calculus is the relation *speaks for*. This is a relation between principals. Let us suppose to have two principals, saying that *A speaks for B* means that if A says something, the other component of the system can believe and infer, that B says the same. The relation *speaks for* implicitly assumes that whatever is said by somebody speaking for you, it is as if you said the same thing. This implication is clearly too strong if we want to represent delegation of responsibilities. In their framework, for delegation to take place, requires two steps. First A has to delegate B something. Subsequently B must explicitly accept this delegation, in such a way that B does not speak on behalf of A by mistake but he must accept and know that when he speaks for A, he is entitled to do it. In the logic if B acting on behalf of A says something, this means that both A and B must contribute to the statement, so both A and B must consent on that, and this is exactly what we do not want with delegation of responsibility. The logic must be extended in order to express this other type of delegation.

I introduce here another, even wider, definition of delegation. Where delegation is defined as the process whereby a principal, authorise an agent to exercise some rights on her behalf for a specific period of time. The novelty of this definition is that there is no mention that the grantor has to hand over any right to be able to delegate, but it is sufficient for her to have just responsibilities. With this form of delegation, the grantor possess only the right to enable delegation, The grantee possess her own rights, but just the combination of the two, can generate delegation. An example of this type of delegation in the real world can be represented by the purchase of a house. In this case the buyer has to go to a solicitor to write down the contract. The buyer cannot write it by herself because she does not have enough competence. However the solicitor has her own rights that the buyer does not pass to her, but still the buyer needs to enable the solicitor to act on her behalf in writing the contract. The solicitor cannot perform the same action on the buyer's behalf without her permission.

Virgil Gligor: You give the power of attorney to somebody, it does not have to be a solicitor.

Reply: Yes.

Roger Needham: There is a difference here though because there are cases where you give somebody a power of attorney to do something that you could perfectly well have done yourself, if you were not ill in hospital or something like that, or where you delegate something by commissioning somebody who is magic and there are things a lawyer can do that you cannot. You say please do this on my behalf, which is slightly different from power of attorney.

Virgil Gligor: Well maybe in the UK it is different but in the US it is the same power of attorney for both.

Bruce Christianson: But it is like saying please issue my car with a test certificate, I authorise you to do that, I do not have the power to do that myself, but I have the power to authorise someone else to do it on my behalf.

Larry Paulson: But there is a difference in handing out which is the right to use your name in conjunction with that.

David Wheeler: I think this abstraction of delegation has possibly gone too far, missed out the recipient or the user, you can delegate rights to somebody but it does require the co-operation of the user for the delegation to work, and two different users might respect different delegations and so by abstracting away the recipient you are possibly making life more difficult to detail.

Bruce Christianson: I do not think there is anything in what Bruno is saying that prevents you from having a two signed delegation.

David Wheeler: Yes, but you are implying that you can delegate between A and B without the co-operation of C, and I think this is actually not possible unless A and B have already done it or unless C is an environment which is willing to accept delegation.

Bruce Christianson: The question is, what order you have to collect the pieces in and whether all three have to be interconscious at any point.

David Wheeler: Yes, but I cannot delegate something to Stew which you may accept, so you do require the co-operation of the recipient.

Bruce Christianson: At some point.

Stewart Lee: And perhaps somebody else.

David Wheeler: I mean in computer terms the programme has to be capable of accepting a delegation.

Michael Roe: Some systems do delegation by impersonation, whereby what you hand over to someone is a marked right that makes them look just like you to the target and in that case in some sense you do not have to get the consent of the target; what you really did need, is to get their consent to use and be a party of a communications protocol that had this feature.

Bruce Christianson: It is fine if you still believe in transparency.

Reply: Something that I did not specify, just because I did not have time, is that it makes a difference whether I delegate an object associated with the key or a person, it is important to make a difference between the two cases.

Virgil Gligor: It seems that there are, one is the act of delegation, there is an act of accepting delegation by the person who receives it, and also accepting delegated access by the service that's being accessed, for example, some services may not accept delegation at all, I mean delegated access, in other words I am willing to transfer to you my rights for a particular object, for a particular service, but the service will not allow you as being a delegatee, so there are those two things which I'm sure you can do.

Reply: Yes, still I am not saying it is a good thing that the grantee must explicitly accept.

Virgil Gligor: That is in the logic. By the way I have seen no example where that axiom is used, not only in their papers, but anywhere else, the second one, the grantee accepts delegation. In all their examples, they use only the first axiom which is the first definition of delegation. But that is a separate point. The point is that also you have to have the access controls in the system to support this, for example, if you look at DCE on the access control lists on the original design, it was possible to differentiate between a delegated access and a direct access. For example, you would have a different access if you were the direct access source than when you are my delegatee for the same object so in some sense you have to be able to make that distinction as well.

Reply: Yes, that is why, for example, in the second definition I say that, the grantor is not any more able to exercise the rights.

Virgil Gligor: That is a separate issue, I am talking about Bruno's access to this file. Either as Bruno, or Bruno for Virgil. Those are two different accesses.

Reply: Oh, yes, sure. You should have for example, a different key for each kind of access, you require. If I access as Bruno I should use a particular token, when I am acting on behalf of you I should use a different token.

Virgil Gligor: And of course the file system has the right to refuse Bruno for Virgil.

Roger Needham: I think it is the case that, as a matter of what Virgil said about Abadi and other friends never using their scheme, it is because in fact although they talk about delegation to both keys and principals that Bruno mentioned, they only actually meant keys and a key not being a principal cannot accept anything. What they're saying is, the presentation of this key shows you are acting on my behalf, shows whoever is acting was.

Bruce Christianson: But there is an example about who controls the key which they do not explore further and which clearly would be an interesting thing to look at.

Reply: And anyway you agree with me that this meaning of delegation is interesting.

Virgil Gligor: There is an example where the delegatee has to explicitly accept it, if you look at the Kerberos forwarding proxy protocol the delegatee has to carry out some steps in order to receive delegation, you cannot be a passive entity.

Raphael Yahalom: Is not there a potential protocol problem if you require the delegatee to participate. Could not the agent abuse that by assuming the rights without acknowledging them.

Virgil Gligor: No, no, it is a very well defined protocol, for me to transfer my identity to you, my rights to you, I have to not only send you the tickets, I have to send you in some sense the protective key from the ticket in Kerberos, so you have to be actively carrying the steps of the protocol to accept the key.

Raphael Yahalom: But there is essentially a last message which somebody can deny receiving and the question is if one of the party is not in a position to behave opportunistically.

Bruce Christianson: The difficulty with Kerberos is the key that transferred is the key that is actually used to exercise the corresponding right and that is where the difficulty occurs.

Virgil Gligor: And because of that it does not correspond to his idea that the grantor no longer has access. I pass to you ticket and the key and then I monitor what you do, because I have the key.

Bruce Christianson: Yes, that is right, you have got it as well.

Reply: When you look at this bunch of papers implementing delegations usually what they say is: I am the grantor, I give you some rights, and then I pass you the key to exercise these rights. While I would like to express is the situation where I pass you the rights and then you can choose the key to exercise the rights. What I have to do is only to authorise the key to exercise the rights but I do not need to know the key itself.

Michael Roe: You are doing the protocol where the grantee loses the right as Rafi says there is the last message problem, typically you are going to get to the stage where neither party has the rights because we are in this intermediate stage where it is one party has given it up in transit and the other party has not accepted it. Presumably if you are really building it you have some kind of transaction monitor recovery mechanism that can roll back from this state if you ever get in to it.

Stewart Lee: The so called limbo log as contrasted with the deadlock of life.

Michael Roe: I mean the wrong way to design it, is to have it fail in such a way that both parties end up having the right; and this is possible even if the cryptographic primitives are correct.

Bruce Christianson: In actually working out how to do this we are back on familiar ground, the question is what we should be doing.

David Wheeler: I have another slight question, auditing, is auditing orthogonal to delegation or does a delegation with auditing have extra properties?

Reply: Well if you have delegation and auditing, you can eventually verify or track down what happened. I mean, I do not want to agree with the statement of yesterday that you do not need to keep auditing. I think that any security system must have this ability.

Bruce Christianson: But you can actually tie the auditing mechanism into the delegation mechanism by saying for instance I give you David this right which you can use provided you present, every time you make a transaction using this right, a certificate that says your intentions have been logged at this particular server, and the right will not work unless you do.

David Wheeler: That is right, and that is no longer authority.

Bruce Christianson: And then it is no longer authority. There is a similar question about the relation between delegation and authorisation, some people say you want two separate mechanisms but there are advantages in a combined mechanism.

Michael Roe: Also, once you have got that measure of responsibility for something, this generally implies supporting audit mechanisms because if there is no record of what happened at all then it is extremely empty to talk about who was responsible for this action that nobody has any memory of.

Virgil Gligor: You cannot talk about responsibility, in my opinion, without audit.

Stewart Lee: I agree.

Bruce Christianson: But this does not mean that it is not a good idea to check whether people have authority to perform and do things as well.

Abuse of Process
(Transcript of Discussion)

Mark Lomas

Goldman Sachs Ltd.

I'd like to start by saying that at a previous workshop I suggested that it might be possible to combine proven security protocols in such a way that a composition was insecure. All of the examples that I was able to give people said "Oh, no, nobody would ever do that", and so I'd like to thank Peter Landrock for giving me some real world examples of the sort of protocols that people are genuinely proposing. I hope they're proposing them without any ill intent, but others might use them in ways that the designers didn't actually intend.

Before describing any of those protocols I'd like to go back a bit in history and talk about ways you might actually abuse a protocol. We tend to look at a cryptographic protocol, or some other form of secure protocol, look at what we believe to be the designer's intent and then try to validate it to see whether it met the designers' objectives. There is a slight problem with that form of analysis because we can't always be certain that we know the intent of the designer. A very old example of a way you can abuse a protocol is a nice term that the Chinese use. They call it "flying money".

We're familiar with normal money, you go into shops and buy things with it, and we're used to it, we think we know what it's for. What the Chinese, or some Chinese people, have found it useful for is as a method of conveying sums of money in excess of the apparent value of the note. What you might use this for is if you're in a country that has exchange control and you're not allowed to take large amounts of money out, but you can take small amounts of money out. You go to certain nefarious people, you say, "I would like to buy a large amount of money to be delivered in some foreign country", and they say "Certainly!" and they hand you a very low value note. And you say "What can I do with this?" They say, "Well if go into this shop in San Francisco and buy a vase something nice might happen to you". And essentially what they've done is they've marked the note in a way that the shopkeeper in San Francisco will recognise and he has agreed to operate a protocol in addition to the protocol that the original bank note issuer thought was going to be used.

Other examples of ways you might abuse a protocol, coming more up to date: Most of us are familiar with covert channels or Gus Simmons work on subliminal channels, where you can essentially convey information in addition to that which was intended in a set of communications.

I'm just looking around, unfortunately Joan Feigenbaum isn't here this morning, but yesterday she said she was worried about the idea of a $20,000 transaction being sent. I think I should perhaps explain that in my job I worry about slightly larger sums of money. I did an enquiry with my employer and I was told that our average external transaction is about $14,000,000. In our Annual Report for last year I was told that we conducted what we believe to be the largest electronic funds transfer in history, in one transaction we bought one third of BP. That would give you an idea of the scale.

Bruce Christianson: Do the protocols you use scale down?

The reason I'm explaining that is it might give you some idea of the motivation behind attackers and this goes back to a point that Roger made in the opening talk, he said, you should think about how people actually try to attack a protocol.

The point about explaining this is to think about the attacker. The sort of attacker who I worry about has got a very good motivation to attack our protocols because if they can manage to cause one extra transaction to go through our system they can probably retire. For that reason the sort of attacker I tend to think about, although I hope that we don't actually have any, is somebody who is in the design team for the protocol, or they might be one of the implementers of the software that's actually going to carry out the protocol or they might be a system manager with root privilege on a set of systems that are participating in the protocol. To my mind, the person who's sitting outside isn't as much of a worry because they have far less information that they could use to attack.

Unfortunately, I'm not allowed to talk about the protocols we use, for obvious reasons, so I'm going to take Peter's examples. Here is an identification protocol [1]. It's a way of proving knowledge of a secret key without actually revealing it. It's reasonably simple and I hope it doesn't have any great surprises in it. A generates a random number R and then encrypts it under B's public key. It sends an identity which actually is irrelevant for this purpose. In order to prove that the value R was actually known to the person who generated the encrypted portion they also put in a result of hashing R. The reason you do that is that otherwise an attacker might just generate a random number and then claim that it was of this form and then continue. The response is to encrypt R or to say fail. The hash of R commits R.

The use of the encryption in the second part is interesting because if the two parties are face to face it might not be necessary to have that encryption, but the intention there is to convince yourself there isn't an intermediary between the two of you.

There's a slightly different scheme. These are similar sorts of principles. Generate two random numbers R and S, and encrypt them,

Essentially you're proving knowledge of R by essentially broadcasting is in the clear, and supposedly showing knowledge of the key and then you get your response that shows that whoever responded was able to extract S.

Those protocols, if you get the keys right, you can prove to be secure subject to a large number of assumptions about key management. But the problem I'm going to describe doesn't actually relate to key management so I won't go through the intricacies of that.

[1] $A \rightarrow B : A, K_B^+\{r\}, h(r)$
$B \rightarrow A : K_A^+\{r\}$

Here's another standard. It's now an international standard, so it must be right. This protocol is for a different purpose. The purpose here is to convey a secret key, in this case, you take a message, you encrypt it under a short-term key and you then tell the recipient what the short-term key is so they can extract the message[2]. You also wish to sign a hash of the key in order to convince the recipient that they got that key from a particular party. There's an alternative technique also from the same standard where you encrypt a message with a short-term key and you can take the identity of the person sending it and the key, encrypt that and then sign it.

Peter Landrock: We had to throw that in because IBM has that as part of their public key infrastructure. They somehow managed to muscle it through.

Before going on, I'll go back to my hypothetical attacker. Let's assume that they came to me with a protocol, let's say it was that one, except they got the keys right, and they've got a nice formal proof of it and they come to me and say, are you happy for us to use this, and I say well that looks OK to me, I'm willing to let you use that. And they say "We would like to embed this in a physical system so we want to be absolutely certain that you're not going to change your mind, it's going to go into this smartcard and we're going to make thousands of them". I say, "Alright you can put it in your smartcard." Remember I've got the nice proof and I can independently conduct the proof because I don't necessarily trust the proof that they generated.

He comes to me a bit later with ISO 11770 and says, "Well you can't possibly worry about that one, it's an international standard, people all over the world have been validating that and I'm happy for you to do your own independent valuation to make sure that's secure". Fine, I'm happy with that one. And then they say multi-function smartcards are fashionable at the moment, we'd like to put it into the same smartcard as the previous one, and by the way, smartcards don't seem to have very much memory so I'd like to use the same key for the two protocols.

I said that in the hypothetical but I know that there are a number of multi-function smartcards where that was exactly the process they went through. The problem is if you happen to see a message of that form, a perfectly legitimate message from 11770 and you happen to be an eavesdropper E, you can then send a message of this form, as you can extract various pieces of information from the original message, and send it to the recipient who has got the appropriate key to unlock it, and they look at all the headers for this message and they say oh this is an identification protocol. The problem is, if they conduct the protocol according to the standard, you'll get back something that will tell the eavesdropper they key that was used in the other message.

Virgil Gligor: Mark, what you seem to have mixed here is two very separate uses of keys, one is in a store and forward environment and the other one is basically a peer to peer authentication environment. The ISO protocol can get even more complicated if you use it for multicast because then any recipient of the message will have the key that goes along with the message. There reason

[2] $A \rightarrow B : s\{m\}, K_B^+\{s\}, K_A^-\{h(s)\}$

that's the case is because you don't want to encrypt a multicast message multiple times. I don't know anyone who'd actually mix keys in peer to peer protocols with multicast or store forward keys. I mean that's a very grave sin.

Carl Ellison: But he's not doing that, the attacker's doing that. All he postulates is that the card is capable of doing both.

Virgil Gligor: Well no, he is actually using the same key for both types of protocols.

Dieter Gollman: The only key he's using twice is the public key and that seems to be a very reasonable assumption, you get the smartcard, you've got a public key, secret key pair: it's you. Then the legitimate person starts off running one protocol, never in his or her dreams would think of using the other protocol, but the attacker intercepts it, takes out some pieces.

Virgil Gligor: One of the principles that we talked about yesterday in a different context is that it makes sense to separate keys and, even though you can design a protocol where you don't have separate keys as Peter pointed out, you can actually design protocols that are not safe. It's much easier, you have a much easier time, designing and proving a protocol, correct in some way, if you separate the use of keys, one key or one pair per function.

Reply: I agree with that entirely but going back to the model I have of the attacker, the attacker is part of the design team, they actually want to cause you to do something like that.

Virgil Gligor: And they do something very naughty, like that.

Reply: They can do something naughty like that, and they might not take, we have deliberately taken simple protocols in order that it's easy to look at the analysis. However, if you look at a typical funds transfer protocol, the description can be huge. I received a document recently showing a particular funds transfer and it took three pages to describe the fields that appear in the messages.

Virgil Gligor: So the point is that the designer can make subtle errors to his advantage.

Reply: Yes he can.

Bruce Christianson: The point is even if a design says these keys must be distinct, how do you know that they were chosen to be as they were? What forces them to be distinct?

Michael Roe: That's exactly the kind of error the implementer can make. The designer has these two tiny separate supposedly isolated protocols, then the guy who comes to write the firmware that goes in the smartcard implements it so it uses the same quantity for both. Nobody notices because the system works just fine, and then sometime later he puts through a fraudulent transaction.

Brice Christianson: The warnings on how to configure these protocols correctly told him where to look to break into it.

Stewart Lee: It's in fact happened. It happened in some secure chip that was used in a gambling machine in Quebec somewhere, where the designer built into the chip a trapdoor only accessible to him using a particular security key and he was able to win the jackpot, an enormous amount of money. Six months later he won the jackpot and nobody realised he was the designer and he walked away with the money. The mistake he made was he went back three weeks later and did it again.

Virgil Gligor: This is more subtle because it doesn't have a trapdoor.

Roger Needham: And it's also worth remembering Ross Anderson's example of the bank which had such a good security on issuing PINs, it took a very long time to find they issued the same pin to everybody because of a bug.

Peter Landrock: Because it was so safe. And another point is that even though, for instance, the data field may be for time stamps and you may not really check the redundancy there, perhaps it's not vital, and so you can use that in another context where it contains totally different information but you're not checking the syntax.

Reply: And sometimes it might not even be possible, in fact, one of your own examples, might as well go onto the final slide, say here is another way of combining these. The important thing to notice is if you receive that first message you can essentially retrofit it into the protocol where you're expecting two random numbers, R and S, and it turns out that although one of the numbers in that message is the identity of one of the participants, if you're expecting a random number, you'll accept it as a random number. The fact that it happens to say Mark Lomas at Goldman Sachs would be completely irrelevant to you, you'd just say oh that looks just as random as any other stream of bits because you don't know what to look for.

Now, one nice observation, again due to Peter, is that we don't actually have to be as strict as you might think in order to protect against that kind of attack. Now the obvious knee-jerk reaction is to say never share keys, but in fact you can fix both of those just by adding a hashing operation to one of the messages, so you don't have to actually change the amount of information sent, you don't need to change the amount of key material you keep in your very expensive piece of real estate you keep in your smartcard. So what I'm really saying is you should do a very careful analysis of the whole system and how different components might interact and how the attacker might cause them to interact in a way that you didn't quite expect.

Peter Landrock: You hinted at it, but I would like to elaborate a little. There are examples where you would like to be able to use the same key pair for different protocols, and it would be nice then to have two that would enable us to find out whether mixing in two protocol is perfectly safe or not. A couple of years ago the government in Denmark had serious problems about issuing every citizen with a citizens' card that would have a public key pair and then reckoned that that could be used for almost anything. I was of course warning against this situation here but what favours that you could use the same key pair is that you don't have to issue each user with three certificates but only one certificate. So it is interesting to see what happens when you mix internationally acceptable cards.

Bruce Christianson: We're used to the notion of including a hash of some of the key when we're doing key agreement to check that you and I are really looking at the same master key. What you're suggesting is that we should do the same thing to establish that we're not using the same key.

Virgil Gligor: This also has value when you want to disclose something to your enemy. Your enemy could get some mechanics right, for example, the US wanted to make sure that some of the missile technology that the Soviets were using was right, particularly the one that disables the firing system, so the US leaked out the whole control system to the Russians. And it had to be a genuine design, I mean it had to come from the US, it had to be authentic, at the same time the Russians could get it by breaking some secret code, so you make mistakes occasionally to your advantage in that sense. You give things away.

A New Concept in Protocols: Verifiable Computational Delegation (Position Paper)

Peter Landrock

Mathematics Institute, Aarhus University
`landrock@cryptomathic.dk`

1 Introduction

Since the introduction of the concept of digital signatures based on public key techniques, a number of additional features have been added, and many new protocols have emerged.

Later, a new concept in identification protocols based on public key techniques was introduced, namely that of zero-knowledge identification-, or proof-, schemes. Out of this grew a number of interesting practical protocols, such as the Fiat-Shamir scheme, which is based on one particular family of digital signature schemes, namely Rabin/RSA, but produces signatures of its own, which are weaker in a sense than signatures produced by the underlying Rabin/RSA pair (see below for an explanation of "weaker"). The underlying zero-knowledge scheme proves possession of the digital signatures on, say k publicly known messages.

To explain our goal, let us consider a slightly different realisation of the Fiat-Shamir scheme, which, as far as we know, has not been addressed before, even though it is so obvious.

Indeed, the owner of a public key pair, (P, S) may produce at set of signatures $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$ for $i = 1, 2, \ldots, m$ and *delegate* these to an executive $E(i)$, who would then be allowed to generate Fiat-Shamir signatures by means of these secrets. It would be understood that k and m are small compared to the modulus n of P, perhaps of the order of magnitude $loglogn$.

The public key of $E(i)$ consists of the original public key P and the corresponding messages $m_{i_1}, m_{i_2}, \ldots, m_{i_k}$ which somehow could be connected to the identity of $E(i)$ through a known syntax, to minimise the size of the key. It immediately follows that this scheme has the property that

(1) The owner can impersonate an executive, but not visa versa, but different executives cannot impersonate each other

So far, so good. We notice that on the face of it the executives have no correlation, i.e. we know of no way by which sufficiently many executives by co-operating together may impersonate the owner, even if we choose larger parameters k and m, as long as $km << n$.

We have emphasised on the digital signature aspect in the above discussion, but we might as well have addressed identification protocols only, i.e. the owner would have access everywhere, but not a delegated executive. Of course the messages tied to $E(i)$ could then determine the access rights of $E(i)$. Before we go on, we would like to stress that property (1) can not be achieved e.g. by means of secret sharing. If the secret key d of the owner is split into two shares d_1 and d_2, from which d may be constructed, it does not necessarily follow that knowledge of d implies knowledge of d_i, but even so, in secret sharing schemes, we do not inherently have any means of verifying publicly, that $E(i)$ knows d_i. We may of course add special protocols as building block of a large scheme which would remedy this, but basically, this seems to require that each executive has his own individual public key pair. We are thus interesting in one further property than e.g. in [BG], where receivers of a share may verify that indeed he got a share, but may not prove to a third party that he has a share from which the original secret may be reconstructed.

Instead, we make the following general requirement:

(*) An executive does not have an individual public key. His public key consists of the public key of the Owner and his *rank*.

2 A New Concept: Verifiable Computational Delegation

The aim of this paper is to introduce a new concept in identification- or digital signature- protocols, namely that of *verifiable (hierarchical) delegation*. As we are reporting on work still in progress, we will concentrate on identification only.

Let us start with a very simple model, where we have the Owner and 2 executives of rank 1. In addition to property (1) above, we want the property

(2) The executives may impersonate the Owner through co-operation

More generally, we could introduce a ranking pyramid system, where the Owner E is an executive of rank 0, who has two executives of rank 1 ($E(0)$ and $E(1)$) such that the triple $(E, E(0), E(1))$ satisfies (*), (1) and (2). In general, an executive of rank $i, E(j)$ - where j is a bit string of length i - would then have two personal executives of rank $i+1$, $E(j_0)$ and $E(j_1)$ such that the triple $(E(j), E(j_0), E(j_1))$ satisfies (*), (1) and (2), *and*, at the same time, all executives of one particular rank through co-operation may impersonate E.

Rather than arguing general abstract nonsense, which may not be realisable, let us explain how we can build a system with one owner and two executives of rank 1. We will then suggest how to build more general systems.

Although we would hope to be able to develop this theory independently of the underlying public key system, it is still a good idea at this early stage to keep track of what appears to be achievable.

The next section presents a new somewhat surprising result in elementary number theory, which will allow us to introduce what appears to be secure hierarchical delegation around some RSA public key systems.

3 A Generalisation of Pythagoras

T.H. Hardy gave the following theorem claimed by Fermat (but proved by Lagrange, after Euler had failed to do so) as an example of beautiful mathematics:

Theorem 3.1. *Let* $p = 1 \bmod 4$ *be a prime. Then there exists integers* a, b *with*

$$p = a^2 + b^2 = (a + ib)(a - ib)$$

In other words, p is not a prime in the Gaussian ring $\mathbf{Z}[i]$, the integers extended by the square root of -1, but the product of two primes. Moreover, the non-ordered pair (a, b) is unique, corresponding to the fact that $\mathbf{Z}[i]$ is a unique factorisation domain. For more on the historical background, see [Gol].

We call (a, b) $(= (b, a))$ a *reduced Fermat pair.* Its convenient to choose notation so that $a > b$ always.

Cornacchia's algorithm from 1908 (see [Coh]) offers the following: Let $p = 1 \bmod 4$ be a prime.

- *input* x such that $x^2 = -1 \bmod p$
- *output* (a, b)

(In fact, Cornacchia's algorithm is more general. We shall return to this later). In other words, if we can find a square root of $-1 \bmod p$, then we can find (the) reduced Fermat pair of p.

What happens if we replace p with any number n? Notice that if $gcd(y, n) = 1$, then

$$x^2 + y^2 = 0 \bmod n \Leftrightarrow (x/y)^2 = -1 \bmod n$$

Any pair (x, y) satisfying this is called a Fermat pair of n.

Note: It is elementary to classify those n which have a Fermat pair, and this was done by Fermat (see [Cox]). Observe that if n is odd, then $n = 1 \bmod 4$. In fact, the question is only interesting if for any prime divisor p of n we have that $p = 1 \bmod 4$.

We have managed to generalise this substantially as follows:

Theorem 3.2. *Let n satisfy that any prime divisor is* $1 \bmod 4$.

1. *The following are equivalent*
   (a) n has a Fermat pair
   (b) n has a reduced Fermat pair
   Furthermore we may define an equivalence relation on the set of Fermat pairs by

   $$(a, b) \sim (c, d) \text{ iff } gcd(ad \pm bc, n) \text{ is not a proper divisor of } n$$

   Note that this is equivalent to $gcd(ac \pm bd, n)$ not being a proper divisor of n.

   The equivalence classes are called Fermat classes. The Fermat class containing (a, b) is denoted $[a, b]$.
2. *Each Fermat class contains a unique reduced Fermat pair. This reduced Fermat pair may be constructed from any Fermat pair in the class as indicated in the algorithm below.*
3. *Let (a, b) and (c, d) be two Fermat pairs of n. Then the following are equivalent:*
   - $[a, b] \neq [c, d]$
   - $1 < gcd(ad - bc, n) < n$
4. *There is a $1-1$ correspondence between the Fermat classes of n and factorisations of n into mutually prime factors.*

We shall not give the proof, which is fairly straightforward except for 2) which is the generalisation of Cornacchia's algorithm for primes, here, but refer to [L] for more details. The perhaps surprising consequence is that it is just as difficult to find a Fermat pair of n as to find a reduced Fermat pair.

The key observation is the following: Let (a, b) and (c, d) be two Fermat pairs of n. Then

$$(a/b)^2 = (c/d)^2 = -1 \bmod n$$

and thus

$$(ad/bc)^2 = 1 \bmod n$$

whereas $(ad/bc)^2 \neq 1 \bmod n$ As we observed above, there is a $1 - 1$ correspondence between Fermat classes and elements of order 4 in the multiplicative subgroup of the integers mod$n$ whose square is -1. If we multiply two square roots of $-1, x$ and y, we get an element of order 2, unless x equals $\pm y$, and as is well-known, an element of order two modulo n yields a factorisation of n into mutually prime factors.

Indeed, if we set

$$SQ(-1) = \{x \bmod n | x^2 = -1 \bmod n\}$$

and

$$E = \{\tau \bmod n | \tau^2 = 1 \bmod n\}$$

then E is an elementary abelian group of order 2^r, where r is the number of different prime divisors of n. Furthermore, if $\alpha \in SQ(-1)$ then $SQ(-1) = \alpha E$, a coset of E in $(\mathbf{Z}/n\mathbf{Z})^*$, the multiplicative subgroup.

So the only problem is really how to go from a Fermat pair to the associated reduced Fermat pair.

It turns out that Cornacchia's algorithm works in general - but the proof which we omit does not. The algorithm is given in the appendix.

Again, so far, so good. Why the headline of this section then? We admit it has nothing to do with the apparent subject of this discussion, but as a typical academically inclined author, we cannot resist: Given any solution to Pythagoras,

$$a^2 + b^2 = c^2$$

we immediately notice that (a, b) is a Fermat pair of c. It then follows that c has a corresponding reduced Fermat pair,

$$p^2 + q^2 = c$$

which was known to the Babylonians, who parameterised all Fermat triplets. Of course, the equation we consider here is the more general

$$a^2 + b^2 = kc$$

for any k, and this justifies the heading. It follows that in order to factor an RSA modulus n, which is the product of two primes each congruent to 1 mod 4, is equivalent to finding two different right-angle triangles, the hypotenuse of which is n, and the other sides have integer lengths.

4 The Core Protocol

Let $n = pq$, where $p, q = 1 \bmod 4$ are primes. The n has two exactly reduced Fermat pairs, (a, b) and (c, d).

Note: We typically need $2log(n)$ bits to describe a Fermat pair, but only $log(n)$ bits to describe the corresponding reduced Fermat pair, which is one of the justifications for the discussion in the previous chapter.

We believe that

1. It is difficult in general, given n as above, to find a Fermat pair of n
2. It is difficult in general, given n as above and a Fermat pair, to find any non-equivalent Fermat pair

So much for religion. The facts are that

1. If factoring is difficult then at least 1) or 2) is difficult.
2. If 2) is easy, we can settle the supposedly difficult question of the Fermat primes, i.e. which numbers of the form $2^x + 1$, where x is a power of 2, are primes.

We shall assume for the rest of this note that the former 1) and 2) above are difficult problems. We then have the following information ranking:

Level 0 (p,q)
Level 1 (a,b), (c,d)
Level 2 n

We now have an obvious basic cut and choose Identification Protocol, which goes as follows:

EXECUTIVE	**VERIFIER**
Choose t randomly	
Calculate $t^2 \bmod n$	$\rightarrow t^2 \bmod n$
	$\leftarrow$ Choose 0 or 1
If 0	$\rightarrow t$
If 1	$\rightarrow \alpha t$

where $\alpha \in SQ(-1)$, represented by (a, b).

The verifier then finally checks that the last received value squared is $\pm$ the received value of the first step.

The discussion of the previous section shows that we do in fact have the possibility of naming two different delegates (a, b) and (c, d), who, if they co-operate, may recover the primes p and q, and thus we have reached our goal. However, additionally we have that our general requirement above is satisfied.

It is also worthwhile noticing that if we use reduced Fermat pairs, it suffices to store one of the numbers a and b, as the other can be calculated from the public modulus then. On the other hand, if (x, y) is a randomly chosen Fermat pair, assertion 1) above asserts that given just x, it is in general impossible to determine y.

5 Hierarchical Delegation

We shall only hint how this idea may be generalised.

Let $p, q = 1 \bmod 2m$ be primes, and $n = pq$. We may then for each $j = 2, 3, \dots, m$ consider the set of elements

$$SQj(-1) = \{x \bmod n | x \text{ raised to the power } 2^{j-1} \text{ is } -1 \bmod n\}$$

It is easy to see that the cardinality of $SQj(-1)$ is $2^{2(j-1)}$. Identifying always x and $-x$, we thus have 2^{j-1} essentially different elements of order 2^j such that the 2^{j-1}'th power is -1. The question now is how to build a hierarchical structure as proposed in the beginning. It turns out that it is not just enough to consider elements, whose order is power of 2.

Example 5.1. Let $m = 3$. Let β be an element of $\mathbf{Z}/n\mathbf{Z}$ of order 2^3, such that $\beta^4 = -1$. Set $\alpha = \beta^2$.
We want two executives of α, x and y. Choose $x = \beta r \bmod n$ and $y = \beta r^{-1} \bmod n$ for r some number. This is all very well. But nobody can verify the identity of x without knowing something about r, say $r^2 \bmod n$. Our first thought was to choose r as some suitable element of order two, but this makes the executives too powerful.

A better possibility seems to be to choose $r^2 \bmod n$ as a message which indicates the rank of the executive. It is then easy to modify the protocol above to allow for identification of the executives of α just introduced.

6 Generalisations

It seems a pity that one is confined to choose n to be the product of two primes both equal to 1 mod 4. A more general approach would be to use Pell's equation

$$a^2 + Db^2 = 0 \bmod n$$

This may be solved iff $-D$ is a quadratic residue. Again this was in fact studied by Fermat. Just as above, we may introduce Fermat classes w.r.t. D, which we call the discriminant: We call a pair (a, b) satisfying the above equation a Fermat pair, and we define two Fermat pairs (a, b) and (c, d) to be equivalent iff $gcd(ad \pm bc, n)$ is not a proper divisor of n. Notice that it never follows that (b, a) is a Fermat pair if (a, b) is, unless $D = 1$. Again, we will call a Fermat pair (a, b) w.r.t. D reduced, if

$$a^2 + Db^2 = n$$

However, it is not true that a solution always exist. Nevertheless, we have the following (see [Cox]): Let

$$p = a^2 + Db^2 \;\; q = c^2 + Dd^2$$

Then

$$\begin{aligned} n &= pq \\ &= (a^2 + Db^2)(c^2 + Dd^2) \\ &= (a^2c^2 + D^2b^2d^2 + D(b^2c^2 + a^2d^2) \\ &= (ac + Dbc)^2 + D(bc - ad)^2 \\ &= (ac - Dbc)^2 + D(bc + ad)^2 \end{aligned}$$

and thus n is of that form, too.
It is known e.g that there are solutions for $D = 2$ iff $p, q = 3 \bmod 8$, and for $D = 3$ if $p, q = 1 \bmod 3$.

A The Algorithm

Proposition A.1. *Let x, y be positive integers with $gcd(x, y) = 1$ such that*

$$x^2 + y^2 = 0 \bmod n$$

for some positive integer n, and assume

$$ax + by = n$$

for positive integers a, b, where $x > b$ and $x^2 > n$. Then

1. $a^2 + b^2 = 0 \bmod n$
2. $a^2 + b^2 < x^2 + y^2$

Proof. Elementary, but tricky. □

This is now used as follows:
Given a Fermat pair (x, y), go to the Fermat pair $(x/y, -1)$. Set $z = x/y \bmod n$. Thus $z^2 + 1 = 0 \bmod n$. If $z^2 < n$, we are done, so assume this is not the case and use Euclid:

$$n = qz + r$$

It follows that $z^2 + 1^2 > q^2 + r^2 = 0 \bmod n$.
Repeat the process with $n := z$ and $z := r$ and keep track of the decomposition of n as a linear combination of the last two remainders.

Example A.1. $p = 29, q = 37, n = 1073$. Then $191^2 + 1 = 0 \bmod 1073$

$$\begin{array}{lclcl} 1073 = 5 \cdot 191 + 118 & 1073 & & & \\ 191 \;\; = 1 \cdot 118 + 73 & = & 5 \cdot (118 + 73) + 118 & = & 6 \cdot 118 + 5 \cdot 73 \\ 118 \;\; = 1 \cdot 73 + 45 & = & 6 \cdot (73 + 45) + 5 \cdot 73 & = & 11 \cdot 73 + 6 \cdot 45 \\ 73 \;\;\; = 1 \cdot 45 + 28 & = & 11 \cdot (45 + 28) + 6 \cdot 45 & = & 17 \cdot 45 + 11 \cdot 28 \\ 45 \;\;\; = 1 \cdot 28 + 17 & = & 17 \cdot (28 + 17) + 11 \cdot 28 & = & 28 \cdot 28 + 17 \cdot 17 \end{array}$$

$$1073 = 28^2 + 17^2$$

References

BG. M. Bellare & S. Goldwasser, *Verifiable Partial Key Escrow*, Proc. 4th ACM Conference on Computer and Communications Security, April 1997

Coh. H. Cohen, *A course in Computational Algebraic Number Theory*, Springer Graduate Texts in Mathematics 138, Springer 1993

Cox. D. A. Cox, *Primes of the Form* $x^2 + ny^2$, Wiley Interscience, 1989

Gol. J. R. Goldman *The Queen of Mathematics, A Historically Motivated Guide to Number Theory* A. K. Peters, Ltd., 1998

L. P. Landrock, *Fermat pairs*, *to appear*

A New Concept in Protocols: Verifiable Computational Delegation (Transcript of Discussion)

Peter Landrock

Cryptomathic A/S

I used to say that the only thing I can trust is what I can prove, but that was when I was a mathematician and now I'm a cryptographer (at least part-time) and then of course that no longer holds. I have to trust for instance that factoring is difficult. It reminds me of a conversation we had about a year ago in the Danish Security Council. Somebody was saying: you know, you don't know what to trust these days, I'm very careful when I'm on the phone, I never say anything important on the phone. And I said I'm much more cautious, I simply never say anything important, you never know.

My starting point is actually the last paragraph of Bruno's article where he wrote "we're actually working to redefine the definition of the logic that we criticised here, to represent delegation of responsibilities. We are implementing this concept using different cryptographic mechanisms. We suspect that there's no particular cryptosystem that can offer solutions or features that other cryptosystems cannot provide", and I have discovered a feature that I think cannot be provided by other cryptosystems but we will see about that. And towards the end, coming back to trust, I will introduce a little three party protocol involving Mark, Roger and myself.

Let me start by an example to explain. Suppose you start with an RSA public key pair. Now in 1986 Fiat-Shamir introduced a beautiful zero-knowledge identification protocol, the purpose of which was to allow users to identify themselves. You'd have a trusted sender who would define an RSA modulus and then they would give identity so to speak to each participant based on this public key pair. But you can use it in a different way, you can always develop a zero-knowledge protocol into a digital signature protocol and this is how the Fiat-Shamir protocol works. So you start with your RSA key pair, p and s, you then introduce k messages where k is a security parameter, identifying, let's call him an executive, E_i. Here are the messages $m_{i1}, ..., m_{ik}$ and they can for instance indicate the identity of the executive and then just concatenate a number, I don't want to get into a discussion to what extent a name identifies a person. Then you sign these messages and this is now going to be the secret key of the executive, these k signatures, so the executive can then sign any message, represented by a hash value h, provided h has exactly k bits: he chooses a random number, encrypts that random number under the public RSA key and then he takes the product of exactly those signatures for which the corresponding bit in the hash value is 1. Then he multiplies that by this random number. That's the Fiat-Shamir signature. It's very interesting that someone patented a variation of this, where he didn't have the random number, and if you don't have the random number you can very quickly recover secrets from a number of protocol runs, and that's just another example of the fact that there's no limit as to what you can patent.

Now we used this system because many years ago when I started Cryptomathic in Denmark, it was really only an attempt to avoid the heavy Danish taxes, just for fun, and then we started developing an electronic banking system for Danish banks and we wanted them to do RSA. But it took more than one minute to get a RSA signature on an 8086 chip running 86 MegaHertz so we suggested that even though we assigned each user an RSA key pair, we'd just use Fiat-Shamir signatures where we could cut it down to a few seconds. And it then occurred to me that if I'm the master, so to speak, I have an RSA key pair, and I could then have some delegates, some executives, I could simply calculate some signatures and then I could allow them to sign using Fiat-Shamir signatures. They could represent me. Then they would still be dependent on my public key pair and they couldn't sign on my behalf because even if you can make a signature using this key you could not make an RSA signature. So here we have the concept of computational delegation where the master has an RSA key pair and he can name any other executives, give them some precalculated signatures and then they can sign.

So, if we look at the properties here they are as follows. The owner of the public key pair may impersonate an executive but not vice-versa, the executive cannot impersonate the owner of the RSA key pair, and of course different executives cannot impersonate each other.

Now I would like to point out that this feature here will not be achieved if, for instance, you share the secret key by means of secret sharing. There you each have a share but you can't really do anything with your share because your share in itself just looks like a random number where with Fiat-Shamir you can actually prove something. And I'd like one further general requirement, I'd like the public key of an executive to simply be the public key of the owner, of this master, and then additionally the rank of the executive, I'd like to perhaps build a hierarchical structure.

And then we are interested in one more property, namely that the executives together may impersonate the owner, so I might say that I have two executives and if they are involved together then they may actually generate a signature on my behalf, an RSA signature. That's again a property that we don't have with Fiat-Shamir.

So, I want to then evaluate on an example, where I have one normal master and he's going to name two such executives. How can I build such a system. Well we have to go way back in history and look at some mathematics that's connected to Cambridge. Fermat had this fundamental result, or he claimed to have it, he didn't give the proof of course, that's usual, take any prime $p = 1 \pmod 4$, then there exist integers a and b so that $p = a^2 + b^2$ and it's really because you have a factorisation in the ring of quotient integers. This pair is unique, Euler was not able to prove this either and it wasn't until Lagrange came along that he discovered it so this is, or it used to be, fairly tricky. Given any number, not necessarily a prime, if that can be written as a sum of two squares then we call that pair a reduced Fermat pair. There's a very nice algorithm from 1908, Cornacchia's algorithm, the paper is in Italian actually: for p a prime, input a square root of minus one modulo p and then the algorithm will output the reduced Fermat pair. And we notice of course that what we then have is certainly the sum of two squares $a^2 + b^2 = 0 \pmod{p}$ and this of course is equivalent to having an element of order four, namely a/b is a square root of minus one. So a reduced Fermat pair gives you an element of order four.

What happens now if we replace the condition that p be a prime with the condition that p is any number. And we now say that the pair x and y is a Fermat pair of n if n divides $x^2 + y^2$. It's elementary to classify the n that do have a Fermat pair, for instance if n is odd then n has to be one modulo four. And it's only really interesting to consider examples where all the prime divisors of n are one modulo four. I don't want to go into that.

Now we've discovered the following result. The following are equivalent: n has a Fermat pair, and n has a reduced Fermat pair.

So if n divides the sum of two squares then n can be written as the sum of two squares. And there's a one-to-one correspondence between reduced prime pairs and mutually prime factors of n, simply given by the fact that if you have one reduced Fermat pair and you take a different reduced Fermat pair, you just want the greatest common divisor pair and you can factor, so it's tightly connected to factoring. I'm not going to claim now that I've found a new way of factoring and therefore RSA is no longer strong!

I will explain a little more about that but we can actually define an equivalence relation on the set of Fermat pairs of n, sums of squares divisible by this n, you say that those two Fermat pairs $(a, b), (c, d)$ are congruent if and only if the great common divisor of this number $ad - bc$ and n is equal to one. That really means that they define the same element of order four modulo n. And we call those Fermat classes.

So what I said earlier was that each Fermat class contains a unique reduced Fermat pair and if you take two different reduced Fermat pairs then the greatest common divisor of this number and n is a divisor of n, so that's just a reformulation. It's actually not very difficult to prove. The only thing that's tricky to prove is that if you have a Fermat pair then you have a reduced Fermat pair. Cornacchia proved that for primes, but it actually holds for any number. And the observation that we're going to make use of here is that if you have a Fermat pair, so n divides $a^2 + b^2$, then of course $a^2 + b^2 = 0 \pmod{n}$ so a/b will have order four. Therefore if you have two different elements of order four, the product is an element of order two which is not minus one, and we all know that those give factors because if you have an element of order two that's not minus one, then it's plus one modulo certain prime divisors and minus one modulo all other prime divisors.

OK I'm almost done with the mathematics, and we can get back to our protocols. The key to the discussion then is the square roots of minus one. Now if you take the set E of all elements of order one or two, that's an elementary

Abelian group of order 2^r where r is the number of different prime divisors of n. And if you take just one element α which is a square root of minus one then the set of all square roots of minus one is simply this co-set αE. The only tricky part of all I've told you is to get from a Fermat pair to a reduced Fermat pair and I just don't have the proof of that.

So I'd like to get back to the protocols. Take an RSA modulus n which has the property that both primes are one modulo four. Then by what I've just told you, this n has exactly two reduced Fermat pairs, so n can be written in two different ways as the sum of two squares, not counting the symmetry. OK I also notice that if you just have a Fermat pair then you'd typically need $2 \log N$ bits to describe it but if you have a reduced Fermat pair then you only need half as many bits. So it makes sense to go for a reduced Fermat pair.

Now how can I use that. Well leaving trust and proof and switching to religion, we believe that it is difficult in general
(1) given an RSA modulus n to find just a single Fermat pair of n, and
(2) given an RSA modulus n and a Fermat pair to find a non-equivalent Fermat pair.

Joan Feigenbaum: Can you prove anything about the relationship of reducing one of those to the other.

Reply: Yes, there are some facts here. If factoring is difficult then (1) or (2) must be difficult because if I can solve (1) and (2) then I can factor so one of them definitely is difficult. OK which one can we be sure of is difficult.

Wembo Mao: If you can solve (2) then you can factorise.

Reply: Yes, but you know the difficult part might be to get the first Fermat pair right. Maybe if I'm given one Fermat pair then it's easy to find the other one, so maybe this is the real difficult part.

Now I think I can convince you that this is very difficult and I actually have Andrew Wiles' word for that, you know, the chap who proved Fermat's Last Theorem.

Virgil Gligor: You say that you can prove that.

Reply: Yes he said this, and I would like to point out that if (2) is easy then we can settle the famous problem of Fermat primes, namely if you are having a number of the form $2^{2^n} + 1$, is that a prime or not, we know that it's a prime for $n \leq 4$ but nobody was ever able to give a general proof that it's not true for $n \geq 5$. So in other words if you can convince us that (2) is easy then you must be able to solve the famous problem of Fermat primes. So that's the closest I can get to an answer to you Joan.

Now, if (1) and (2) are difficult then we can introduce delegation, I mean computational delegation. We have the following levels: here is my public key, that's the public level, here's my secret level where I know the primes p and q, and here's my delegation level where I give each executive one Fermat pair, it doesn't matter if it's reduced or not because of my theorem, and I know I have exactly two Fermat classes and I give each of my delegates a pair from each and I prefer to give them a reduced Fermat pair because that minimises the bit length. Now I can imitate Fiat-Shamir, I can for instance start with a standard identification protocol, a cut and choose protocol, and I have a delegate here I have to verify. The delegate chooses a random number r and sends $r^2 \bmod N$ to the verifier. The verifier chooses a bit, if the bit is zero then the delegate must return the initial choice r, if it's one then he must return α times r where α is an element of order four, that he can construct from the Fermat pair: remember if I have a Fermat pair (a, b) then a/b is an element of order four. And you could actually also go on and make xxx the second two protocols because you can do that whenever you have an identification protocol. Then you may ask yourself, how we can carry this further. Why should we stop at a elements of order four. The world lies in front of us.

Let's assume that we have a xxx set example mod p and a xxx set example mod q of order 2^r. I would of course remind you of Wembo's talk yesterday. I have to constrain myself to elements whose order modulo p and modulo q is the same. If not, if I exhibit any element whose order modulo two different primes is different then I can factor using that element. And that's why it has to be a square root of minus one, it has to lie on the diagonal of Z/pZ and Z/qZ.

So I can find then an element α with $\alpha^{2^i} = -1 \pmod{n}$, where $n = pq$. And then I can look at the various powers here, so imagine you have elements of order sixteen, eight and four and if I then give α_i to an $i-$star general, that's just to make the point, I can generalise the protocol from before in the obvious way where the $i-$star general proves that he possesses an element of order 2^{i+1}. By the way, this identification protocol also proves that any prime divisor of n is one modulo four, that's a little subtlety.

So there seem to be a lot of possibilities here. However, it's not as nice as I thought, because if I have minus one here and then I have two different square roots of minus one, these are my two executives at the first layer, then I can go up to elements of order eight, but of course anybody who possesses an element of order eight can square and then possess an element of order four, so this is definitely an executive of higher rank than him, yet the two of these can represent the master, so I don't get quite the hierarchical structure that I'd like to. And I think perhaps the way of getting that is by also assuming, that I have all small prime divisors r of $p - 1$ and $q - 1$ for instance an element of order three. And that's an interesting connection to Wembo's talk, as I'm really going for the complete opposite of the property that Wembo was talking about.

I think I should explain to you how Cornacchia's algorithm works, then it looks very nice. The key to the whole thing is the following proposition. Take two natural numbers x, y which have greatest common divisor equal to one, and assume that the linear combination $\alpha x + \beta y$ is equal to n. Now we can prove that if $x > \beta$ and $x > \sqrt{n}$, and if furthermore (x, y) is a Fermat pair, then so is (α, β). And that's a smaller Fermat pair, $x^2 + y^2 > \alpha^2 + \beta^2$, so I'm using reduction here. How can I use that then. Well, given a Fermat pair (x, y) of my modulus n, then I first go to this Fermat pair $(x/y, -1)$. So I can assume that y is minus one, so $x^2 + 1 = 0 \pmod{n}$. And then first of all, if $x < \sqrt{n}$ we are done, otherwise we use Euclid. We divide x into n, $n = qx + r$, and since $x^2 + 1^2 = 0 \pmod{n}$, my proposition tells me that (q, r) is also a Fermat pair and my proposition also tells me that $q^2 + r^2 < x^2 + 1$. And then I just keep going, and keep track of what is happening and finally I end up with the remainder as one of the reduced Fermat pair, and the other number is the co-efficient of the previous term. So if you just think about it for a while you'll see that it works.

I thought it was better to give you an example: suppose my primes are 29 and 37, so n is 1,073. Here I have a number of order four, the Fermat pair $(191, 1)$. How do I then construct a reduced Fermat pair. Well, I divide 191 into my modulus and get the remainder 118. Now I divide 118 into 191 and if I keep doing that I end up, the first time I have a remainder whose square is smaller than n, I stop, and *voila*, $28^2 + 17^2 = 1073$. Previously only known if the number to start with is a prime.

There's a chapter in my paper called Generalisation of Pythagoras because what I've just demonstrated to you can also be considered a generalisation of Pythagoras' $a^2 + b^2 = c^2$. What does this tell you, it tells you that (a, b) is a Fermat pair of c, I know it's a very particular multiple, but still the multiple of c is written as the sum of two squares. What does my theorem state then, it states that c is the sum of two squares, and that's the way that the Babylonians actually parameterized solutions to Pythagoras, that c can be written as the sum of two squares. And incidentally this is then the difference of the two squares and this is twice the product of the two. So you may say that it's just a generalisation of Pythagoras as well.

OK, even though there is some trust involved in what I've just demonstrated to you there's also a proof. I mean I did prove that I can get from a Fermat pair to a reduced Fermat pair and I did prove that if both of those two questions I was asking are easy then I can factor.

I would now like to show you a protocol where you don't know the proof and I am going to take advantage of that. And just to warm you up here's the flavour: I'm going to discuss protocols that appear to demonstrate unwarranted delegation.

First we have a public key pair which is an RSA public key pair, n is a product of two primes. We all know that if anybody knows $p + q$ then from the modulus and $p + q$ you can factor because you can easily recover p and q. Right, so if anybody has an RSA pair like this, I would like to convince that person that I know $p + q$. How? Well here's the protocol. I say, I know $p + q$. Now I won't just send it over the Internet because then everybody will know. So what I'm saying is send me a random number, r, and then I will return $r^{p+q} \bmod n$. I can do that. Why can I do that? It looks very impressive doesn't it, it really looks as though I know $p + q$, I mean we all know about the discrete log problem and what have you, but there is nothing to it. Because you know that $n = \phi(n) + p + q - 1$, where ϕ is the Euler function, and if I raise any element to the $\phi(n)$ then I get one mod n. So in other words $n + 1 = \phi(n) + p + q$ and therefore if I take any number r and raise that to $n + 1$ modulo n, then that's $r^{\phi(n)} = 1$ times r^{p+q}. So, after all I didn't know that much but it looks very impressive. I have another variation where I can prove that I know $\phi(n)$. Give me any number and I will raise it to $\phi(n)$.

Wembo Mao: What is the verification of this protocol.

Reply: Well Alice can verify because she knows $p + q$ too, I just want to convince her that I know I know $p + q$.

OK, now here's then a real example that, as I said, involves Mark, Roger and myself, and I call this protocol How to Blackmail Goldman Sachs over the Internet. Goldman Sachs publishes the RSA key n with public exponent 3, this is for digital signatures, perfectly safe, everybody does it, Europay, MasterCard, what have you, and n is 1024 bits, because Mark has told me that this is open. I write a letter, week one, to Goldman Sachs, to the top manager, I say send me £1 or else I will publish the first byte of your secret exponent. What happens. Nothing of course, I'm sure they get letters like that every day. Here's my next letter, I say here's the first byte of your secret exponent, send me £2 or I'll publish the second byte. So there's a little worrying going on, Mark is called in and Mark looks at this. And first of all he might say well I can't even get access to the secret exponent so I don't know if this bloke is correct or not. But let's just assume that he can, and he discovers that, gee it's correct. In fact I can do that, I can do that, we can try it next week Mark.

Bruce Christianson: But he'd get 256 of these letters a day anyway so they're still not worried about it.

Reply: That's right exactly, that's the point, it's only eight bits so he was lucky.

So third week I say here's the second byte, send me £4 or I'll publish the third byte. And now, even Mark starts getting worried because they don't get 65,536 letters every day. So what does Mark do? He calls Roger. In fact he keeps calling Roger every week for the following months as I send more and more bytes and finally Roger asks: how much does it cost to replace the public key that Goldman Sachs is using to certify all Goldman Sachs International customers. Because that's really what it's used for and you know it's not trivial to replace a certification key, we've been involved in evaluations around that for very large institutions, and it may easily cost you a million pounds. But let's just say that they don't have so many users so the answer is £65,537. And then of course Roger says, pay him week 16 at the latest, because then the cost incurred by preventing this is smaller than the cost by going through till the end. So that was Roger's philosophy yesterday. Now I can do this, you all test me, generate an RSA public key pair next week, use public exponent three, say 1024 bits, and I will send you every week throughout the whole year, 52 weeks say, I will send you one byte after another of your secret exponent.

I challenge you to find out how I do that, this is very much in the spirit of Fermat of course, and if anybody attended my Rump talk at Crypta last year then he will know the answer of course. But I can do that so it's a challenge for you to find out how.

Mark Lomas: What would you say if I said my public exponent happens to be on your slide. That the exponent I use is on your slide, I don't use 3.

Reply: I could probably do it for five but I can't do it for ....

Bruce Christianson: For 65,537.

Michael Roe: This is one of the reasons why people use that.

Reply: But it does mean that RSA public exponent 3 is less secure.

So do you take the challenge, I can tell you the solution tomorrow, or do you want to have the solution now.

Bruce Christianson: What's the difference in price?

Carl Ellison: You lose the fun of finding it for yourself.

Bruce Christianson: I just wanted to check I don't have to pay double tomorrow.

Reply: I'll give you one clue, if I give you my published public key n, say 1024 bits, you only have to determine p or q, which is half as long, or $p+q$. In a way the 1024 bits only contains 512 bits of hidden information, that's the hint.

Now I've just gone through this algorithm with you to recover those two numbers. Now look what happens if I then carry out the process among the imaginary numbers. So the one factor I have of a multiple of 1073 is really this complex integer $191 + i$ because this times the complex conjugate is $191^2 + 1 {=} 34 \times 1,073$. So I take $191 + i$ times 5, subtract that from 1,073 and end up with the number $118 - 5i$. What I have this time with the norm of this element is that $|118 - 5i|^2 = 13 \times 1,073$, but it represents a smaller Fermat pair. And then I divide $118 - 5i$ into $191 + i$ and I end up with remainder $73 + 6i$ and again 73 and 6 is a Fermat pair, $|73 + 6i|^2 = 5 \times 1,073$, and I keep going and I finally end up with this remainder $28 + 17i$, which is the Fermat pair I was looking at. And if I then continue then you'll see that exactly the same numbers occur again but swapped, see this 191 occurs down here now as a coefficient to i, $5 - 118i = 5(1 - 191i) + 1,073$. So this process here is reflected. It's not that I can say, oh yes but this is obvious because so and so and so, it's just a very funny observation or a very striking observation and I can't explain it. I can't prove that you end up with the right number here but that doesn't matter, I proved that one here. It would be nice to see a proof that explains all this, perhaps there's even more going on.

So I apologise for the fact that there's some mathematics involved but my justification is that G.H. Hardy gave Fermat's theorem as an example of beautiful mathematics in his book "A Mathematician's Apology". And I really started working on this two years ago when I was in Utah, but it wasn't until much later that I found finally discovered how to generalise Cornacchia's algorithm.

Delegation and Not-So Smart Cards
(Position Paper)

Bruce Christianson and James A. Malcolm

Faculty of Engineering and Information Sciences
University of Hertfordshire: Hatfield
England, Europe

Abstract. We consider three aspects of delegation: What kinds of binding are required between real world artifacts and bit patterns? What are the atomic elements of a security policy and in what ways must they fit together? What are the physical requirements for dedicated hardware to bear this semantic burden?

1 Bindings at the Edge of Cyberspace

Real world artifacts cannot be authenticated remotely, except by binding them to bit patterns first. Real world artifacts have provenance. Bit patterns do not. The binding between a bit pattern and a real world entity therefore cannot be effected solely by other bit patterns, but requires some real world artifact with provenance, such as a piece of paper or hardware. Trusted, tamper-proof or tamper-evident hardware is always required to effect remotely verifiable bindings. For example, if biodata is used to verify one individual remotely to another, then the hardware which does the verification must be tamper-stop, and must be trusted by both parties, whether or not the biodata is shared. If this hardware is local to one party, then it must be remote from the other. Consequently a remotely verifiable binding between the transmitted bit pattern and the trusted hardware is still required.

Mappings from bit patterns to real-world entities should occur only at the boundaries of cyberspace, not across the middle of it. Bit patterns should be bound to each other via unique bit patterns and not via real world entities. The only unique bit patterns are those derived from unshared keys, whether or not such keys are used cryptographically. To effect the bindings on the boundary of cyberspace, unique keys must eventually be bound to hardware and it's best to do this explicitly and use physical means to manage the hardware. Moving unshared keys from one piece of hardware to another is a serious security risk. So let's not do it. Far better to bind principals to keys via hardware in the first place, and then delegate from one key to another.

Explicit delegation is required anyway, both to represent the security policy and to provide an audit trail. Rigid bindings between keys and physical objects or locations can combine with delegation to allow a greater degree of assurance that operations have been properly authorized and properly carried out.

2 The Truth about Delegation

Delegation should be expressed as a relationship between consenting bit patterns. Rights may be delegated from keys to other keys. Delegated rights must be explicitly accepted. The key used to exercise a right might not be the key used to accept it. Delegated rights may include the right to accept or to confer other rights, as well as the right to exercise a right or to delegate further the right to exercise it. Delegation rights are a form of access right. For example, if R is a right, then the right to delegate R is another right R', and the right to accept R is another right R''. The rights R' and R'' may also be delegated from keys to other keys. Rights should be bound to keys and thence to hardware.

The form which delegation tokens should take (or even the nature of the basic building blocks from which they should be made) is a question which seems to be being debated on the wrong issues. Even the type of delegation which the tokens should represent seems to be a source of confusion: for example many systems do not distinguish between delegation of authority and of responsibility, in spite of the fact that conceptually the two go in opposite directions.

For example: consider an Archive service, a Broker, and a Client. B arranges a transaction giving C some of A's file space, which is realized by the transfer of a number of delegation tokens. Now A is responsible for maintaining the integrity of C's data. (But responsible to whom? To B? to C? to C's principal? Who decides? and how does A know?) Conversely C is authorized to determine the content of A's disk. (But was B ever so authorized? Need A know who C is, even in cyberspace?) Similarly C is responsible for keeping A's disk free from naughty bit patterns such as pornography and unlicenced cryptographic keys (but responsible to whom? to A? to B? to the government of Mordor?) And A is authorized to supply data on C's behalf (but to whom, and under what circumstances?)

Other things which may be delegated include access rights, roles, ownership, identity, policy, liability, or the capability to "speak for" another party. For the moment, at least, we should develop mechanisms which are agnostic about the interactions between these different concepts, and allow the representation of a wide range of policies which we can then explore. In particular, it is not a good idea to distinguish between delegation and authorization at the level of mechanism, since this would require us to decide whether or not two principals are the same in real life by inspecting bit patterns.

We usually think of a chain of delegation as beginning with the owner A of some resource D, and passing through a sequence of intermediaries, each of whom confers some (possibly more restricted) version of the rights over D to the (less trusted) next party in the chain. These transfers may limit the rights in a number of ways, such as restricting them to be used only

- at a particular time
- at a particular location (server)
- in connection with a particular transaction (type, or authorizer)
- when pre-logged by a particular audit mechanism

Most of these restrictions can be regarded as special cases of requiring the capability representing the rights to be

– countersigned by some particular key, or
– used in conjunction with a certain other capability

Since delegations are themselves invocations of rights, they may be subject to similar restrictions about where they must be recorded. A useful technique is to add "back edges" to delegation chains to link them together into a mesh containing cycles. As well as being useful for audit purposes, insertion of such back edges can ensure that we always eventually have a point at which we should have *issuer = verifier*. It is possible to ensure that this condition can be checked remotely, and has the same outcome whether we look at keys or at hardware.

3 Untrusting Shared Resources

We discuss the case of asymmetric keys here. However even this case requires trusted hardware, a reliable audit trail, and the binding of certain actions (such as signatures) to certain physical locations (such as a particular smart card). Consequently the situation for symmetric keys may not be very different.

The conventional model of delegation is that restrictions occur as rights pass from a secure domain to one which is considered less trustworthy (by the owner of the resource), or to hardware which is considered less reliable (for instance more liable to penetration, malfeasance or incompetence.) This view has valuable or 'unrestricted' rights residing on a 'central' mainframe server regarded as 'secure' (eg held in a locked room) and the chain of delegation passes through network computers to a portable PC down possibly to a smartcard at the low (ie 'least secure' point) of the delegation chain. The idea is that hardware which isn't under the control of an appropriately expert regime shouldn't be trusted with rights which are any more valuable than necessary.

We advocate inverting this view. It's much better to think of the smart card as being one of the few processors that we can control absolutely. Expensive resources such as mainframes will inevitably end up being shared across a number of security policy domains for economic reasons. No matter how "well" they are managed, nobody can be sure what such a shared mainframe is up to. The point of a smartcard is that it's a cheap enough resource that it needn't be shared. The valuable (less restricted) capabilities should be located on the unshared resources, and physical means should be used to control the circumstances under which these capabilities can be used (or further delegated.)

Smart cards can be locked up when not in use, and can be configured to work only when in the presence of some other hardware (key) from which they are normally kept apart. Another advantage of smart cards is precisely that they can be used without much peripheral equipment, so it is feasible to maintain physical provenance by requiring use in restrictive conditions, which allow recording (eg video). We tend to fixate on RSA with two prime factors, but more factors can be used. This allows a master key to be kept in several parts on different smart cards in several different physically secure locations. Periodically these sites can be visited in turn to enable the master key to delegate to (sign) a number of medium-long term keys, each of which can be held on a single (physically access-controlled) smartcard, and can in turn be delegated to a short term key kept on a networked PC locked in a filing cabinet, etc.

Further, more restricted delegations can be made to untrusted things like file servers and mainframes (at the low point.) At each stage the delegations made should be inter-logged, and should require the cooperation of several keys. Note that we are not suggesting that central mainframes are *untrustworthy*, merely that they should not, in fact, be trusted. This approach ensures that there is no disincentive to sharing expensive resources across a number of security policy domains. Many current analyses of requirements for a delegation framework take what can be proved (the facts) as primary. We argue that of at least equal importance is a systematic account of what assertions principals were, in the end, willing to make on the basis of faith.

4 The Secret Life of Smart Cards

In this section we discuss the physical requirements for smartcards which are to be used in delegation chains in the way suggested in the previous sections. Again, there are reasons to suspect that these requirements may be more general.

First, smartcards must be tamper-evident. They need not be tamper-proof, since we can (and should) control the physical context in which they are used so that a single break-in cannot be disasterous. But the smartcard must be deployed in such a way that the tamper-sealing will be checked regularly by different parties.

Second, the physical matrix upon which the smartcard is realized must be irreproducibly unique, like a bank note. Otherwise Eve could get the secret out, blow it into a blank card and pass the second (untampered) card off as the first. So this requirement is the same as that for plastic money, but with the smartcard being the bank note passed from hand to hand. It must also be easy to check whether the internal secret (assuming the seal intact) matches the physical matrix. A third requirement is that is should be possible to require smart cards to be physically juxtaposed in some way, like lego, with each other or with more complex hardware, before certain operations are enabled.

Finally, smartcards should be very limited in capacity and power. It is self-defeating to put a Cray on a smartcard, when this becomes technically feasible. An EDSAC would be a much better idea. Little memory leaves no room for trojan horses; slow cpu cycles mean that stolen cycles result in humanly perceptible delays; low bandwidth for communication means even lower bandwidth for covert channels; and so on.

The moral is that slight changes from the present physical requirements for "smart" electronic tokens would support significant advances in security.

Delegation and Not-So Smart Cards
(Transcript of Discussion)

Bruce Christianson

University of Hertfordshire

It occurs to me that some of the people here didn't hear my talk in Paris last year, so I've slipped some slides in from that talk. But I've reordered them and I've changed the ending because the market research we did showed that audiences preferred a happy ending.

Here are some questions that might be interesting to consider in the context of delegation. The first is this perennial question of what bindings do we actually have to make between real world artifacts and bit patterns. If we're going to have delegation what constraints does that put on the bindings we need and the mechanisms we have to do it.

What's the atomic structure of delegation? Rather than trying to go immediately for the single universal sub-atomic particle, what are the actual different types of delegation that we have and how do they fit together with each other and with some sort of security policy. What are the blocks that we have to build this from and how do they fit together, and can we at least come to some agreement about what terms we ought to have the debate on.

And finally what sort of physical and semantic requirements does this place on hardware, in particular if hardware were supporting a properly managed secure notion of delegation what would it look like. I'm going to argue that it would have to be slightly different from how it looks now but even certain slight differences, if they're the right ones, can make you feel a whole lot better.

This next bit isn't intended to be controversial. This isn't the distinction David introduced earlier between people and computer things. This is a distinction between bit patterns and real world entities. I think I do mean real world. For example, people are entities in the real world, names are bit patterns. A key is a bit pattern, a key in a particular smart card is an artifact. The text of a program is a bit pattern. Telnet running on a particular box is a real world artifact. A key certificate is a physical entity, it's a piece of paper with lots of stamps on it, it's in a particular safe deposit box in a particular bank. Electronic key certificates are bit patterns. The key difference between these things is that real world objects have provenance. They have a history, it's not possible for them to be in more than one place at one time, if one's a copy of another you can work out which is the copy and which is the original if you're careful. With bit patterns there's no fact of the matter. Bit patterns do not have provenance, all bit patterns are, in some sense, the same.

And somehow, from time to time, we have to create a binding between a bit pattern and a real world artifact, and those are the bindings that are hard to verify remotely. It's hard to verify a binding between a bit pattern and a real world artifact unless either you, or some artifact which you trust, is in the vicinity of the artifact.

I ought to say a little bit about the baggage that I'm bringing to bear on this question. My interest is integrity in wide open systems. By integrity I'm concerned with two things. The first thing is whether the right thing was done and second thing is whether the thing was done right. And the point is that usually different parties, principals, whatever we want to call them, are responsible for these two aspects. The party that's responsible for authorising the carrying out of a transaction probably isn't the party that's responsible for running the correct methods to achieve that transaction on the right data set. By wide open I mean network resources may be shared, operating systems resources may be shared, they may be shared with people you don't know, which is sometimes not very good, they may be shared with your close colleagues which is usually much worse, but the real security threats are from insiders, in other words people who are authorised to share the resources with you for some purpose just probably not the purpose that they're actually using them for.

So we've got this binding problem and cryptography doesn't help much. Digital signatures will bind bit patterns to bit patterns, which is fine, but they won't bind a bit pattern to a real world object. You need another real world object to do that and it's either local or remote, and if it's local to you it can't help with verifying and if it's remote to you, you've got another problem unless you trust it. And the only unique bit patters are bit patterns which could be used as private keys, whether or not you actually use them as private keys.

There's a kind of ghastly analogy here with the database world. The classic hole you fall down is you end up in security trying to model the model inside the model and the database people have an iron rule about that, they say there's no representation of real world relationships in the model. All relationships connect terms, all terms are keys, some keys have the property that they refer to real world objects, some keys refer to other relationships in the database. But there's no notion of modelling a relationship between two real world objects directly. So one of the things I'd like to toss in here is that perhaps having that sort of attitude towards unique cryptographic keys would help us exercise a bit more self discipline.

David Wheeler: Bruce, can you have a world-wide artifact?

Carl Ellison: Yes, the world.

Reply: That's a harder question than it looks, as usual David. Yes, I think you can.

David Wheeler: Can you verify it remotely?

Reply: Well, alright, the crucial experiment is this. You can build a Josephson junction at liquid nitrogen temperatures now with these ceramic conductors, you can build one thirty feet across, the technology exists. You then discover empirically whether it changes state instantaneously or not, then you know whether relativity is right or quantum theory is right, they can't both be, and then I can answer your question. No-one's done the experiment yet, although the technology has existed for about seven years.

Well, now, access rights, capabilities, credentials, whatever we want to call them, somehow we've got to bind these things to keys eventually, because the only things we can verify remotely are cryptographic signatures made using keys. And here's a bad way of doing it, which is to bind two bit patterns together via a real world entity, and you can solve that problem by making it much worse and binding them together via a bit pattern that's a non-unique name, and it's better if you try and bind principals in some way to keys and keep these bindings relatively rigid and then bind the keys to other bit patterns... I mean I'm not saying anything new here in this audience.

But if you actually want to do anything useful you've got to take these bindings a step further because you've got two loose ends, you've got the physical system that the transaction is actually being run on, and you've got the principal who's authorising the transaction. Typically you want to say that this key can act in this role and can authorise this transaction, and today this transaction is made up of these methods running on today's version of the system which is on this particular piece of hardware. And so you've got somehow to maintain the table artifacts that give you the connections between these different things which is what last year's talk was about, but not this year's talk. But the key point is, Carl's said this hundreds of times, that the bindings between bit patterns and real world entities should happen only at the fringe of cyberspace, not in the middle. And we all say "yes" and then we go off and we ...

So the obvious argument is to say let's bind principals to keys, that's the tricky bit, so let's do it via the hardware. People are very bad at cryptography so the key has got to be in a piece of hardware really, so the obvious thing to do is to bind the key to the hardware that it's in somehow, and then never move the key and use some sort of physical means to control access to the hardware.

Moving unique keys around is a really bad thing to do, I mean if you think about it, it means they're not unique anymore, so let's not do it. Let's say each key is bound to a single piece of hardware forever, and instead of moving keys around we'll use delegation, this magic thing, to move around whatever it is we were going to use the key to do.

The other hobby horse I ought to ride is the *transparency is bad for you* hobby horse. A lot of work has been devoted over the last twenty years to building distributed systems in such a way that you can do processing in lots of different places and it doesn't matter. So you build a system which allows you to run the process on any processor that happens to be free and indeed you won't even know which processor it's running on. In fact, very often, even very careful examination of the audit trail will not tell you and I'm advocating the view that this is perhaps not a very good thing.

But the developments of the last twenty years are not wasted, we can turn them around, and we can say well the requirement that certain operations only ever happen on certain specific pieces of hardware, thanks to all this work on distributed systems, is no longer an onerous burden. It's now relatively straightforward to say no matter what you're doing, where you're doing it, there are certain things, not on-line things, but certain things that happen rarely and at intervals, that it's OK to say must happen on certain pieces of hardware.

Mark Lomas: Sometimes it's not that it has to be a particular piece of hardware but it's a hardware configuration. I had a experience recently of a system which was perfectly OK as it was, they said we'd like to get rid of this obsolete piece of equipment and we'll put two new computers in its place and the new system they were going to replace the old one with is functionally equivalent, all the interfaces behave just the same way, the only problem is, in order to carry out its task, the two systems have to communicate and thereby, I won't go into the rest of the details.

Reply: Well that in a way is a special case of the problem that we're trying to solve. The argument isn't oh gosh we're in a distributed world now we need explicit delegation. The argument is if we had some correct understanding of delegation it will allow us to audit the systems we've already got properly.

So, we've had some discussion already about what it is that we're delegating, are we delegating the right to do something, are we delegating the ability to act in a particular role, are we delegating responsibility, authority, ownership, identity i.e. delegating my identity to someone else. Are we delegating policy, that's where I might say OK I'm going to give access to this resource to anybody who can produce a certificate signed by Joan saying that PolicyMaker said that I should, and a certificate signed by Roger saying he's collected the money. Are we delegating liability. Or are we doing this "speaks for" thing so we're delegating to a channel, or something like that. And the next point I'm going to make is that for the time being the actual mechanisms that we use have to be reasonably agnostic about this. We don't understand what it is we're using delegating to achieve well enough to be able to build mechanisms that are predicated on a particular answer to that question. That's why I'm so strongly in favour of the decision PolicyMaker took, to be purely syntactic.

We have to look at these semantic burdens but we want to try and get into a situation where we're arguing about significance of what we did, not about what we should do.

Well, here's a very simple example. We've got this archive A, there's some sort of broker B who has been given capabilities by the archive that allow the broker to read and write to certain sections of the disk, and a client C comes along and says to the broker I would like you to archive my files for me, and the broker says certainly, and delegates to the client the ability to read and write those sections of the disk. And the question is, what's really going on here. It's clear that A is now responsible for the integrity of C's data, but it's not clear who A is responsible to. Depending on the contractual details, A might be responsible to C, or to B or to some other party, and it ought to be possible by looking at the chain of delegation tokens that are used, to work out which. C now has the authority to decide what's on parts of A's disk, that's an authority that A started off with and that now somehow C has got. Somebody, probably C, but possibly B, or maybe even A, is responsible to somebody else, possibly A or possibly the government, to ensure that the data on the disc is pornography free.

A has possibly succeeded in handing off that responsibility with the delegation token without anybody noticing. And A has now got the authority to supply data on C's behalf to some other parties, but it's not quite clear whom. And again, the chain of tokens ought to make that clear, there ought to be some way of answering these questions. But we're never going to get there if we insist on having one type of delegation with everything then used in a single way.

The other point this example is intended to illustrate is that any naive view saying well delegation went from A to B to C, or delegation went from C to B to A, doesn't quite work out correctly. At each stage there's some sort of exchange of rights and authority and responsibility, but it's a two way process. Delegation really does seem to involve, as Carl said earlier, some sort of consenting action from each end.

Why delegate? Well, users use more than one piece of hardware and I just tried to persuade you that moving keys around with the users is a bad idea, unless you can move the hardware around with the users. But since the user, by moving round, is usually departing from the remote end that he's going to want to verify at some point, you can't have it both ways. And the second point is that security policy domains do not map neatly onto administrative management domains. If you have an expensive part of a processing system that is shared between several different contracts that are being conducted by different teams who are operating different security policies, you somehow have to have policy domains co-existing. And the corollary of that is that users mustn't, as a matter of fact, trust their system managers. System managers may be perfectly nice people, but it must not be the case that you can be shown to have trusted your system manager.

So, the reasons you need explicit delegation: the first is to represent the security policy in order to allow you to work out, *a priori*, whether to authorise something or not, and the second is to provide some sort of unforgeable audit trail, so that you can work out, *a posteriori*, who actually did what, and on whose behalf and why.

I'm going to argue that you do actually need both of these things. People often say it's the certainty of getting caught that deters crime, and that's true up to a point. But there are all sorts of reasons why it is necessary to be able to look back into the past and work out what actually happened. But you really don't want to have to do that very often. If every credit card transaction required somebody to go and look in the file and produce the paper copy, the cost of credit card transactions would be astronomical.

The game is a two part game, we often see this with communication protocols. There's a cheap and cheerful up-front mechanism which works most of the time and keeps the riff-raff out. And then, behind this nice badge-sensitive door-opening mechanism, there's another door with a marine on it with a loaded weapon and somebody else who will check your identity very carefully. The point [of the first mechanism] is to make sure that that second one isn't bothered by a large proportion of spurious calls. So you do need both these mechanisms, they do have to interact and you need to have a delegation mechanism that supports both types.

Mark Lomas: There's also the intermediate case. You might not care what happened you just care how much of it happened. I'm running a credit card business and I say it would cost me too much to investigate fully, but as long as I'm not losing more than one percent of my transactions, I'm going to carry on doing business in whichever country.

Reply: Sure, but again, you probably want to keep the whole audit trail because you'd want to be able to defer decisions about which bit you look at. So potentially every bit has to satisfy the property.

How do we do delegation. Well typically the idea is you have some token that you can use to do various things and somebody else ends up with some other token that can be used to do the same kind of thing but which is restricted in some way. So you can only use the token if you're David Wheeler, or you can only use the token to invoke a particular transaction or you can only invoke a particular transaction on a particular system at a particular time, within particular time limits, or for a particular reason: e.g. you have an authority to invoke a particular method provided that method is being invoked as part of a transaction with particular parameters. Or you say this token can be used to invoke certain operations provided it has an audit certificate that comes with it, that says it's been logged in the right place and that PolicyMaker has said that you can use it in this way.

I think it's also desirable to have a single mechanism that can handle both delegation and authorisation. There's quite a lot of work done on the premise that these are orthogonal issues, and the difficulty is if you treat them as orthogonal, you've got to have a way of knowing whether two keys bind to the same real world entity or not, which is just the thing that we agreed was a bad idea earlier on.

So the conclusion of that is that delegation is a relationship between consenting bit patterns and it really does require the consent of both parties in some sense. But the difficulty is that the key which somebody uses to accept a right, or a credential or whatever it is, may not be the key that they use to invoke it, and indeed they may not be the same principals that are exercising those two facets, your manager may have the power to accept obligations on your behalf.

So the situation is that delegation is just another kind of access right but the right to delegate is an access right, and the right to accept delegated authority or responsibility is also a right, and these other rights can be delegated. And the policy may put constraints on that but the mechanism shouldn't. And this is a really strong constraint. I've tripped over that so many times, that suddenly remote verification requires me to know who someone is in real life. Whereas what I want at that point is to find an appropriate hypotenuse delegation token that says that this key has delegated to that other key so that it doesn't matter whether it's the same person or not.

So, in other words, one of the key things about delegation is that it's a way of avoiding the movement of unique keys and very often you can arrange so that parts of the web of delegation go in a circle. The idea is that if you start with a disk server delegating the power to write to files, sooner or later a request is going to come to read or write a file, and it's got to come to where the physical file is and that's the point at which the final verification is going to be done with the same key because if we made a rigid binding between keys and hardware then saying it's at the same place and saying it's the same key cease to be different questions.

So, we can use rigid bindings, it's not automatic, you have to work harder to get this property, and very often this involves putting back-edges in a long delegation path, and you think well I'm only putting this in to get this nice property of having a circle, but then you look at it and you think well actually these are exactly the kind of edges that an auditor would require me to put into my accounting process if these were people here instead of computer systems.

So we can ensure that that happens provided we have cyclic webs of delegation, but the problem is where do you put the strong capabilities in these circles, where's the point of the circle that you have the capabilities with lots of power, where's the point where you have the capabilities that can just do a little bit, like spend 14p on a stamp or something. And the usual view is you say well you've got powerful mainframes up here, local servers, workstations, personal computers, down to smartcards and genetically engineered amoeba down here. And the usual view is that you put the strong capabilities on these secure, properly managed systems that are in locked rooms up here and you put the little capabilities that it doesn't matter if you lose on these cheap and cheerful smartcards down here.

Now if we think about the economics of it, why are distributed systems around. Well, resource-sharing is the answer that springs to mind. That's always been the driving force behind distributed systems and networking. Expensive resources will be shared across security policy domains, you know that's just a fact, and smartcards are cheap enough that you don't need to share them. When I say smartcards, I don't really mean smartcards, I mean humpty-dumpty smartcards, something that's cheap enough that you don't have to share it with anybody else. And I'm going to argue that the valuable capabilities somehow have to be on these unshared resources, and that the only capabilities that you should allow onto highly shared resources are ones that can't do very much except stuff that you want done with them anyway. So if some bad guy is going to steal your capability and use it to wash your car and take your rubbish out, that's fine.

Of course, I'm not saying that mainframes are actually untrustworthy, I'm not saying that systems administrators are bad people, or that it's not possible to design shared hardware. It's actually quite easy, a little bit of care with the microcircuit design for address space sharing and a few little bits of fiddling on how we do time slicing, and you can have a mainframe where you can share it and nobody can look over anybody's shoulder. What I'm saying is that mainframes should not be trusted. I'm not saying that they're untrustworthy, I'm saying that you should not, as a matter of fact, take anything a mainframe does on faith.

Which leads me to my next hobby horse, this is the *don't confuse me with the facts* hobby horse. We've always got to start by saying well what assertions can we make and what are the factual basis that justify those assertions, what are the facts and why are they true. But I think it's dangerous to stop there because in any case where you've got a significant security policy requirement there are going to be cases where I take the next step in a protocol, not because I know that the facts justify it, not because I have a way of verifying, even after the event, that the facts justified it, but because my security policy tells me that I can. The security policy says, if Joan say's it's OK then it's OK. In other words Joan is trusted over, not whether or not this fact is true, but whether or not I should believe this fact is true. Again, Roger's not here, I don't mean belief, I mean BAN-type belief, I mean what you're willing for your ambassador to say in public.

Virgil Gligor: Is this form of trust transitive, because she does trust SNMP.

Joan Feigenbaum: That depends on the policy.

Reply: This form of trust is not transitive.

And that comes to the crucial hobby horse that I rode around two years ago that says, after you've settled the question what's true and why is it true, I think you've also got to go on and look at the question what beliefs do the different participants in a protocol or a policy have and what is the basis on which they hold those beliefs. Not why are those beliefs true because they might not be, at least when you're doing counter-factual reasoning about one or the other participant in the protocol, the insider who you think might be a bad guy, you've got to make these counter-factual assumptions. And so things like referential transparency go away, you can't use a fact based analysis.

Michael Roe: Can I just jump in on that.

Reply: Of course. We're very nearly using your time anyway.

Michael Roe: There's a way in which mainframes could be trusted from the point of view of composites, that you have this highly managed machine that runs some software and because of the way it's managed, there is widespread belief that that software does what it claims to do and for that reason it does become shared.

Reply: Yes, that is the acceptable way of getting a shared infrastructure. But it's a belief-based mechanism. It's not saying this system is safe to use because the Amoeba kernel is running on every system here and if any system isn't running the Amoeba kernel or the Trusted Computing Base then I am vulnerable. This is saying I have to run a kernel with certain properties or I'm vulnerable, it doesn't matter what you're running since I can protect myself if I have a system with these properties, and it so happens that we've all agreed that the forty-line C code that Simon wrote over the weekend works, so as it happens for our own different reasons we all adopt it. I run it because you run it, you run it because you've checked it, Bill runs it because somebody else has checked it, and we're probably not willing to say in public what our reasons are but we're all as a matter of fact prepared to run the same piece of software.

Michael Roe: I don't trust the result if you run it on your machine because I'm not sure what the physical environment is, but if you run it over here on this thing, we'll all know what it is, I believe the result.

Reply: It doesn't even matter if there's not one thing we can all run it on, so long as each of us has something.

The other point that I made, we've got to put these highly valuable things on these smartcards that people carry round, lose, can drill into with their little brace and bit or read with a John Bull printing set or something, how do we do that, how do we make that safe. Well you have to impose physical constraints on what has to be done with these tokens before they can have a desired effect. I think it's Mark who told me that when a bank wants to send a very large number of bearer bonds through London, what they do is they don't use Securicor, they tear the bearer bonds in half and they give all the right-hand halves to someone, they put him in a taxi, and when they're convinced he's arrived at the other end, they give the left-hand halves to someone else. It's not worth stealing the right-hand halves of bearer bonds. You can do the same thing with physical tokens. We can also do the same thing with keys on tokens. The classic example is RSA with more than two factors but we just had a very nice example of an alternative approach to that from Peter Landrock.

By deploying this and having the right physical management structure to complement the bit management, we all get obsessed with the bit management because we think we can do that, if you have the right physical management as well then you can maintain the provenance of the tokens on the smartcard and if you have a nice big web of delegation where there's all these back edges that you didn't think you needed, then you've got lots of ways of revoking. You've got lots of points at which you can break the link and therefore you don't have this business of having to try to get to the revocation authority before the shops shut on Friday.

Finally, I will stop with the next slide, the secret life of smartcards. The point basically is that you don't want them to have one. You don't want smartcards to be doing things you don't know that they're doing, and heaven forbid but there's a certain amount of evidence to indicate that some software does things you didn't know it was doing, for reasons that you might not approve of if you knew them.

So this idea of saying, well it will be really good when technology allows us to put a Cray on a smartcard because then it will be like the PC revolution all over again but with even less self-discipline. The properties that you actually want, the properties humpty-dumpty wants smartcards to have, are that they should be tamper-evident. Not tamper-proof, I don't care if you drill into it and get whatever's in out, but I want to be able to look at it and tell that somebody has. I want to know that if the seal's intact, the secret's unique. The physical protocol has to mean that they're checked regularly, they're not sitting in a bank vault with nobody looking at them, but they're only bought into a white environment where they can actually be used under very peculiar conditions.

The physical matrix of a smartcard must be unique and the binding to what's inside must be externally verifiable. Current smartcards do not do this. I take your smartcard, I drill the secret out of it, I get a blank smartcard, blow the secret in to it and palm the second one off as the first one, you can't tell, the tamper-evident seal is intact. And finally we want smartcards to be stupid, we want them to have just enough memory to do what they're supposed to do, just enough CPU cycles that if they're doing anything else we'll be able to measure it, low enough bandwidth that the bandwidth of the covert channel is about one bit per fortnight, and so on. I'll stop there.

David Wheeler: You did, on one of your slides, write the two words unique keys, that struck me as rather interesting, were you implying public/private or were you implying pairs of keys, one of which is a copy, and which.

Reply: I'm certainly not implying keys one of which is a copy of the other because then there's no fact of the matter. I am not necessarily saying there's a public key and a private key.

David Wheeler: There's usually two keys, one coding one decoding.

Reply: In the case of public key then my concern is that the private key must be unique, in other words there must only be one copy of that bit pattern in existence. There are other cases where there is a symmetric key which is derived from a one way hash of the unique key, and the symmetric key is shared, the unique key isn't.

David Wheeler: In that case is the shared key unique.

Virgil Gligor: No, it's a shared secret.

Reply: The shared key isn't unique but the secret from which it's derived is. And there are cases where the secret, the unique thing, or what's derived from it, are what we would call a nonce, or a confounder, or a challenge, not actually a key, but my argument is you can regard them as keys in a database sense.

Mark Lomas: It's also sometimes useful to send messages to yourself.

Reply: Oh yes, particularly if you're moving from place to place.

John Warne: Your use of the term transparency. I refer back to the Ansa project which I worked on, I want to bring up that then transparency didn't have that interpretation. Essentially when you actually formed the binding between things you could dictate where those things were but you didn't want to know them on every invocation to those things, but you could subsequently interrogate that if you suspected something is wrong.

Reply: Yes, that's right. When I say transparency, different groups of people have interpreted that word to mean different things. The usual interpretation is that you can't see the internals and have no way of seing the internals. Now Ansa used the word almost in the exact opposite sense.

Stewart Lee: What you mean is invisibility rather than transparency.

Reply: The thing that is what I mean is the bad thing.

Stewart Lee: Transparency means you can see it but you can see through it. (Laughter.)

Reply: I don't mind, the point is there's lots of good systems out there where people have designed upon the basis that it doesn't matter where it's done, and indeed it's important that you can't find out, whereas what this actually shows is that it's important to have anonymous proxies if there are cases where any audit trail must lead back to the company secretary and it mustn't be possible for the vendor to find out who actually authorised the purchase.

Certification and Delegation
(Transcript of Discussion)

Michael Roe

Centre for Communications Systems Research
University of Cambridge

OK, so what's the difference between certification and delegation? It's well known in this business that you can't tell what a protocol is supposed to do just by looking at what cryptographic transformations get performed on bit patterns. You can have the same crypto protocol being used in completely different real world situations to do different jobs. In particular, many protocols have got something that looks a bit like this, where a private key signs some stuff that includes a field that identifies something and vaguely tells you what it is, a public key, a validity period, a serial number and some data — again being very vague about what actually that data is. And then whenever implementors come and see that, they say "Yeah, X.509 certificate, got a piece of code that already does that, call that subroutine." It's particularly bad because now in version 3 of X.509, there is this very convenient bit bucket, whereby you can define any new extension with whatever semantics you feel like and stuff it in there.

Now the consequence of this is that not all X.509 certificates have the same meaning. There are lots of different things being done with these certificates which are completely different jobs and have completely different associated semantics. So the question of what actually is a certificate just really doesn't make sense. You have to ask for a particular application. In particular, these objects can be doing delegation. We can do something like this: I set up a server which provides some kind of service. The right to grant access to the service I've just provided I can delegate to a particular key, K_A. Whoever has the inverse component of that can use it to sign some X.509 certificates for another key K_B, which then gets used to sign the invocation of the service. This is a delegation of access rights from one key to another key, and I've said absolutely nothing about real world identity in this. But nevertheless we could all still be doing this with a protocol that looks syntactically like X.509.

What am I really supposed to do about real world identity? Well it's a funny thing, the original standard is very abstract and one of the first things that tried to put flesh on the bones of this was Internet Privacy Enhanced Mail. OK, it's died the death now and is lost in history but some of the things it tried to do in profiling X.509 been followed by lots of other people. And one thing you might note about that is it places particular emphasis on organisational certificates. These used the names of somebody in the context of their employer, and there was all this discussion about when you move employer you get a new name; when your job role changes you get a new name and all the certificates with your old name get revoked. And this really ought to have been a clue that it was not about personal identity. It's more like: the XYZ corporation's key delegates some right the corporation has to — I've said here public relations officer — some employee who is currently acting in a role of that organisation. So it's delegation of a right from one key to another. If you just look at X.509 then it all looks the same and so you say, "What's the difference? CA certificates are just the same!" But signing that certificate for the XYZ corporation's public key wasn't delegation in the same sense, because there was nobody who had the right to delegate. There was nobody who had together all the rights for all the companies within the country and could give them out. This was why partly, although corporate CAs were very easy to set up, as it was clear who had the authority delegated in the first place, national CAs were extremely hard, not from the technical point of view of implementing the software that produces the certificates but from the point of view of deciding who is responsible; who has authority to issue these things? The answer is that there is no-one because this isn't delegation. The other case was delegation — even though they both looked exactly the same from the point of view of the bit patterns that had been sent along the wire.

On a related note, there are residential certificates with street addresses mentioned in Privacy Enhanced Mail. Nobody ever issued them, again precisely because it's totally unclear who has authority to sign one of these things. Also I think you should be very aware that using an organisational certificate for non-organisational purposes is a very dangerous thing. Using your certificate you've got in your role as an employee of your employer and then using it to talk to your lawyer about a transaction to sell your house or something like that that's unrelated, puts you under a certain risk. This risk didn't matter if you were using it for what that key was supposed to be used for, because the person who delegated that right had the right to delegate it to someone else anyway. But if you suddenly start misusing certificates for purposes other than that what it says in them you raise this risk of misbehaviour — what if the CA does something slightly unexpected?

Stewart Lee: Also it puts your employer at a certain risk.

Reply: Indeed so! I mean it cuts both ways.

Returning to the discussion of physical keys, I have here a physical key and its got a name, it calls itself YR1035. And the thing about keys is that they open locks. They're handed out to people to enable them to open doors. Typically, when you employ someone you hand them out one or more of these things and it lets them do things that are part of their job. And well, keys may have names but nobody really knows what their names are. The lock certainly doesn't care that this key is called that.

Bruce Christianson: Or that you are not.

Reply: Or that you are not. It's just physical possession of this key that opens the lock and that's all that matters.

And many of the things that are actually built using public key technology have this same property. They don't try and do anything like non-repudiation; they don't keep memory of the digital signatures, of what the access was. Something with the right bit pattern turns up, the key fits the lock and something happens. And so that's why that kind of delegation of rights view of what a certificate is, is a good match for what is currently being done with this kind of technology.

Bill asked earlier who's responsible for misuse? Has this changing view of what a certificate is, shifted who is responsible from the certificate subject to the certificate issuer? Well, if you didn't have non-repudiation this is a kind of non-question. The protocol is saying nothing about responsibility or liability, you have to have some entirely out of the system agreement on who is responsible. Who gets the blame is essentially uncorrelated with whose fault it was. And you may think that this is not a nice thing for the protocol to do to you, but this is the pain you have to take for having decided you didn't want a non-repudiation protocol.

So, the moral is that this kind of model of authorisation on keys works even if you don't have non-repudiation. If you decide you want to bolt non-repudiation on afterwards and build all that additional mechanism, so you can in addition work out what really happened and whose key it was, it doesn't do you any harm. The additional non-repudiation function doesn't break things that already existed. However, if you take this strict personal identity model and are basing your whole view of the world on the assumption that we can track down someone's personal identity, and then you discover we can't because we don't have all this additional infrastructure that we hoped was going to be there — then you're in trouble. So this model is a better starting point, given that you know a lightweight implementation is the best and any kind of non-repudiation service isn't going to exist for the foreseeable future.

That's the main thing, but to add a slightly related observation: We now have applications which sign executable code, things like signed Java applets or Authenticode which does signing of windows binaries. When you do this you get into a trust management issue. Somebody out there, whoever they are, has signed this bit pattern and is proposing that I run it on my machine. Should I do this or not? There's a question! If they gave me a bad program then I'm going to come to some harm — possibly. There's an analogous issue in the key distribution problem — they give you a bit pattern which is allegedly a key and if I use it cryptographically for some purpose and it turns out to be bad then I might come to some harm.

So it could well be all this work that's done with PolicyMaker and trust management things — somebody gives you a bit pattern, do you act on it or not? — can be brought to bear. We can solve both of these problems in a unified way. I was about to say does it make sense to delegate an access right to a programme and I think it does. I mean certainly that's old technology in single system cases, where you put the *setuid* bit on a binary and this gives some property to that particular piece of executable code. But you can equally well do it in a network case where you attach some digital signatures, use some key to bind some properties to a particular bit pattern that's a program. That gives that program some magic access rights to do something on some particular type of machine.

Bruce Christianson: Well the devil's in that last phrase isn't it?

Reply: Oh, I was very careful to say that. Yes. Authorising something to have some right over everyone everywhere, doesn't conceptually make sense. It makes sense within the context of some particular target system – which brings us back to your talk.

Stewart Lee: So you don't believe in God.

Reply: No, but the CA advocates do.

Carl Ellison: Provided they get to be God.

Bruce Christianson: The argument is you mustn't bind a capability to a programme text, only to a process.

Reply: Yes, you can't say that this programme text regardless of it where it's executing has this right. The binding is to the bit pattern, but what binding a property to the bit pattern means is that when this bit pattern gets executed in a particular place it acquires particular rights. Be a bit careful there not to say that the signature binds anything to the real physical rights.

Bruce Christianson: This is exactly the qualification people ought to make about electronic key certificates, but never do, as well.

Carl Ellison: I just wanted to add one side-note. You mentioned the residential certificates that never get issued. We also never saw the organisational ones.

Reply: I've seen some organisational ones but only typically within corporate networks, and they didn't get between organisations.

Carl Ellison: I believe I know why we don't see them outside the internal networks. That's because at least in every sizeable organisation I've worked in the employee list was company confidential and so they are not about to release their certificates for their own place.

Reply: And also there's the issue that for them to be meaningful outside, there needs to be this other layer above that gets one corporation's public key to the other. We never built that.

Bruce Christianson: And what never happened either was that corporations never intercertified. They could have stepped out and used a PGP type approach, but the whole thing was irrevocably wedded to this X.509 database.

Carl Ellison: There's a single root.

Reply: Yes. I mean now the CA salesmen are arguing that you can't do that. You have to go through this root, and sometime even the vendors of software have colluded in this by shipping software that because of the way it's constructed it only lets you do it this way, despite the fact that from the point of view of logical and practical reasons it would have been perfectly sensible to do exactly what you said.

Joan Feigenbaum: I think IE is shipped with VeriSign root certificates.

Carl Ellison: Both IE and Netscape were shipped with a number of roots and I believe both of them allowed you to add your own. But they don't well document how you do that.

Joan Feigenbaum: And the number of people who have done it is minuscule.

Reply: There are also subtleties to do with sending a secure message with Netscape. You're required to have a path from you up to the "roots" before it will let you actually send encrypted e-mail messages. And of course the "root" of the sender has to also be a "root" of the receiving end — this can be very bad news indeed.

Joan Feigenbaum: So their answer to that is they ship these default roots so that as soon as you sit down and want to send signed e-mail to somebody, or encrypted e-mail, you can. I find this very bothersome because, yes, arithmetically you can — but what have you achieved by doing that if you and he have never thought through any kind of delegation policy? Which you haven't of course.

Bruce Christianson: We're agreed about the bit pattern it's the semantics that are in dispute.

Joan Feigenbaum: Yes, exactly, so I think that they've really got it completely backwards.

Audience: It depends on what your service is, when you get a certificate with a root that you don't have among the defaults, it gets added to the default ones, and you just need to add the new one.

Audience: Will it accept me as a root?

Audience: As far as the software goes you can add it.

Audience: Yes, but if you get mail from someone who has a certificate from me and that mail is signed, what are you going to do?

Reply: That's the other problem, what you would like is to be able to say, yes, I know it's his key and I will accept that key just for signing messages from him. The software does not give me the ability to do with that. The only ability I have with the software is to give away the ability to do virtually anything to my computer. I feel somewhat reluctant to do that, and the only other alternative I'm allowed is that both of us give the authority to do anything to our computers to VeriSign.

Joan Feigenbaum: So I have a very general question about this configuration management issue. These things get shipped with some default roots. I think we seem to think in this room that that would be a bad thing; we shouldn't do that; everybody should have to install his own policy which presumably he's thought through. So the last time I discussed trust management at an informal workshop it was with a bunch of people from philosophy and psychology, not with people from computer sciences and engineering, who knew a little bit about computer science and engineering. So here we're in sort of the opposite situation, we've got a bunch of people who know a lot about computer science and engineering and are sort of a bit philosophical. We're having a pretty philosophical conversation here I would say as computer science goes. Alright, so what those people said was they all liked the idea of the individual rights point of view — I should control my own computing environment and Carl should control his own computing environment, and each corporation should be its own policy domain, with a bunch of sub-domains. But they said, like the middle of the second day, "This is really complicated, I don't want to have to configure every piece of software that I buy." As soon as one person had that insight the whole room just started nodding.

Carl Ellison: So Joan, that's a business opportunity.

Raphael Yahalom: Not necessarily.

Joan Feigenbaum: I noticed that.

Raphael Yahalom: It's a bad business decision because what Joan is describing I interpret as being the typical marketing reaction: saying we don't want to configure. I can perfectly identify with that attitude for the simple reason when you walk into a bank (and we have a session about that later) you don't go behind the scenes and investigate how they operate your money, you somehow rely on other agencies without any very well defined policy. Why do you trust the National West or go to another bank? For the same exact reason, you don't want to go into a great deal of trouble in trying to deal yourself with different authorisation of CA roots and so on. You prefer that somebody would do it for you, providing you think it is a reasonable sort of assumption. What makes it a reasonable assumption is a tough question but you certainly don't want to do too much work yourself.

Stewart Lee: I think we've been here before today when Peter Landrock said never say anything important.

Virgil Gligor: Actually to pick up on Rafi's point, on Joan's point - in everyday life we implicitly trust an awful lot of things, and basically we want to sort of separate concerns. You trust your bank, you trust your medical doctor without verifying whether or not he's taking the right tests. You trust, trust and trust. I don't really think that's a fundamental problem. The fundamental problem is that we don't know that in the first place we trust things. In other words we don't know what the exposures are. Often we don't want to know what the exposures are because if we don't then we have plausible deniability, among other things. So the first thing in trust, in my opinion, is to find out what it is that we are trusting. Second point is, why you are trusting those? The why is very, very, complicated very often. Now, as far as the marketing is concerned, probably 95% - 99% of the people are not interested in what they are trusting and they're not interested in why they are trusting what they're trusting. So these companies that are shipping certificate chains with roots must know something.

Joan Feigenbaum: No I'm not sure they do.

Raphael Yahalom: So, what you are suggesting is that when you when you walk into Heathrow Airport or JFK you will have the right to go and inspect the plane yourself rather than rely on the ASM who can?

Joan Feigenbaum: No, let me clarify. I'm not suggesting, I'm reporting a market demand. I want to also pick up on something you said. You said that in the course of real world life we trust a lot of institutions like banks and restaurants and doctors and this and that. I agree with that completely and I think that's necessary, but I don't think there isn't a "well defined policy". I think there is a very well defined policy somewhere and that is the difference with the electronic world. We don't. It's not that we haven't personally thought through policies, it's that nobody has thought this through. There's a lot of spontaneous trust going on.

Dieter Gollmann: Coming back to our pseudo-philosophy discussion. I think what you saw was not just a blip, it's an indication of what I say is the fundamental dilemma in computer security. You have security unaware users with specific security problems. How do you get out of this trap? The nicest example to look at is the history of evolution criteria. You start off with the Orange Book — very rigid — people keep complaining that it's a solution for the military and they want force it on everybody. For more freedom, you move to European criteria (ITSEC), which says assurance and function are separate dimensions: we will allow you to say anything you want about your product and we evaluate it and we give you assurance. Then the obvious complaint is, what is the average company customer to make out of such a certificate with weird claims but high assurance. It's not useful and you see the pendulum swing back already. There's a Federal Criteria now and the Common Criteria, and they say we have to find some sort of protection profiles. But I haven't seen anything happening much in the protection profiles world — for example rephrasing the old Orange Book criteria so we're compatible with the past. That's where I see the major challenge for computer security, to find these reasonable solutions which security unaware users can deal with. It can't work if you give out a complicated security product. The customer would say configure it, sort out what you really want. It doesn't work.

Bruce Christianson: But the undesirable situation is where you have a mechanism that smuggles a policy in. A mechanism that's completely neutral and gives people this horrific burden of freedom is one in which I immediately go to someone that I trust and say "What should I do?"

Joan Feigenbaum: I would disagree that you can't have complicated policies. I think you can't expect everybody to come up with his own complicated policy. What you can expect people to do is to trust some well known complicated policy person.

Virgil Gligor: Generally we want to use the defaults of a very sophisticated system.

Reply: The thing is the vendor of the system is often supplying you with a policy built in to the spec of a protocol (or its implementation), and in fact when you look at it you discover it's an extremely undesirable policy to have.

Carl Ellison: That's the problem, or that's one of the problems and we see that. The thing I would add to the scenario of the security unaware user is that as far as we've been able to tell, users fiercely demand to remain security unaware. They want no involvement with security whatsoever, which includes no penalties — I mean no security violations. They just don't want to be hassled, they don't want to know about it, they don't want to do anything about it, period.

Stewart Lee: That's not quite my experience because in my view users want to have a very small number of quanta of security from none to top. Maybe there's one in the middle. My model of this is to have a gear shift on the side of the computer that moves me from one to the other and a sort of quality of security (QoS) measure that can be set. The difficulty I get into is that a lot of the things we're talking about here are very valuable amongst us, but they're totally incomprehensible to the typical user and unless we can render them comprehensible — and that means make them very simple and very encompassing — we won't get anywhere.

Joan Feigenbaum: Why does the typical user in a computer system have to understand the full policy? That's not true in the real world. A typical user doesn't understand.

Stewart Lee: He doesn't. I agree absolutely. That's what I was trying to say, that the typical user when he's sending e-mail to Mum will set the security level at very low. When he is doing his personal tax return he will set the security level as high as he thinks he needs to set it. We might even charge him, the higher the security the more he pays.

Reply: I would suggest that the PC model of the world is perhaps to blame here. You're saying basically every user is their own system administrator, so they're in charge of all configuration management, all installation and because of this all security policy. This is not the position you want to be landed in.

Stewart Lee: But nevertheless it is the position we have been landed in because it is the position of computers. People have personal computers.

Audience: Could we go back to this issue that Bill raised about trust, where we talk about uncertainties? By and large in some sense the policy that we associate with that is observation. What we do is if we find that one of those uncertainties turns out to be a certainty that we dislike, then we no longer trust. It's that kind of association: we can't predict the outcome of this thing, but we implicitly trust it, but we have to observe this thing.

Reply: The whole CA world is setting itself up for a big fall precisely because of this. Nobody's been hurt badly yet, but when somebody gets hurt badly you won't know why, and you won't know exactly who in this big mess was actually at fault. All you will know is that this thing that was supposed to protect you and you trusted, didn't work.

Stewart Lee: What you're saying really is that the CA world hasn't had its DC-10 incident yet.

Reply: And it's due real soon now.

Stewart Lee: When it does, everybody will eschew that particular supplier's certificate.

Reply: Well the problem is that there is a bigger danger that they may just eschew all certificates, and everything else that was involved in that whole process that caused them to get hurt.

Stewart Lee: People when they are dealing with something that goes from something they trust to something they don't trust have a sort of mob psychology, they just absolutely leave.

Reply: Well, that's right. In particularly, we know about the list of CAs configured into browsers, but most people don't know how to change this. So if one of them misbehaves. . .

Stewart Lee: Most people don't even use it.

Bruno Crispo: If you are using one of the CAs that is embedded in your browser you know this.

Reply: You typically don't know this.

Carl Ellison: If I'm connecting up to a secure web page, I do not know which CA.

Bruce Christianson: You don't know which root was actually used to certify that page.

Virgil Gligor: You can find out.

Reply: If you're a knowlegable user who knows how to do this.

Carl Ellison: It took me many, many months to find out.

Stewart Lee: Can you find out if you are connecting to a secure web page. Is that obvious?

Carl Ellison: There's a little indicator down some place on the border of the form.

Bruno Crispo: Embedding the CA root in the browser could be dangerous for the user but it's much more dangerous for the CA themselves. Because I'm embedding my root key in a piece of software. If I revoke my key I'm in trouble because it means I have to change all the codes and software. So it's dangerous for the user but it's much more dangerous for the CA to embed his own certificate in the browser.

Bruce Christianson: I think the question is when it happens are people going to say you're not going to get me on an aeroplane or are they going to say you're not going to get me on a DC-10?

Carl Ellison: Well, there were people who said both.

Stewart Lee: Absolutely right, and they never sold another DC10, not one.

Discussion Session: Differences Between Academic and Commercial Security

Virgil Gligor, Peter Landrock, Mark Lomas,
Raphael Yahalom, and John Warne

Virgil Gligor: Ralph Carmadie who is now the Head of the Sloane Foundation and former VP for research at IBM had some interesting statistics about research in general. He points out that 92% or thereabouts of all research is actually industrial research that has direct applicability to products and only about 8% of it is government and academic type research. In the view of that when you say research wouldn't you say that this is academic and government research as opposed to industrial research because if you mean all research the ratio there has changed tremendously.

Even in industrial research much of it goes into that shaded area today because usually the industry other than IBM doesn't do research that's done for the sake of research. For example, one company that does very little research up until very recently, is Microsoft, almost zero research for the largest part of their existence so far. And I mean by that zero academic research. But they've done a lot of product oriented research and that's why those things have to be better.

Stewart Lee: Do you accept that various players in this scheme of things have different meanings of the word research?

Virgil Gligor: Well, this is what I'm trying to figure out, what research means there and what applications mean there.

Joan Feigenbaum: I totally disagree with your view of industry, I've spent my whole career after grad school in an industrial research lab which is absolutely not the way you just described.

Virgil Gligor: I included you in IBM.

Joan Feigenbaum: Oh, well, that's a very interesting category! Anyway the point is that somebody in some management work chart thinks that the bulk of what's going on in this industrial research lab is applicable.

I'm sure there's quite a few people in the AT&T work chart who think that too, but if you look back in retrospect on everything that's done and compute how much of it actually found its way into a useful product you'll see — it would be great if 14.4% of everything we did was to buy, that would be fantastic.

Peter Landrock: I want to elaborate then, by academic research, it doesn't matter whether you are employed by a university or an industrial lab, what matters is if it's academic work that is published.

Dieter Gollman: Yes, I think that is a major point, published research.

Peter Landrock: I don't care who your employer is.

Virgil Gligor: It is supported by the way in the US we have rankings of the universities and in about 1986/87 the top ranked university in the terms of research was IBM.

Joan Feigenbaum: So maybe we should say basic research as opposed to academic research.

Virgil Gligor: Well, maybe.

Stewart Lee: Let us move on to the other members of the Panel.

Raphael Yahalom: For the last two years I have been spending my time, quite a lot of my time, too much of my time commuting between Boston and New York working with a few financial institutions in New York and doing research at MIT Sloane School of management. In particular my main romance was with CitiBank, Citicorp, and that started thanks to a person named Mark Levine, who about three years ago managed to get into from St Petersburg into the cash management systems of CitiBank and transfer a few million dollars into accounts that he opened in California and Finland and in Israel. A couple of brief points about this incident. (a) Mr Levine as far as I'm concerned could have easily brought down the bank and done another Barings, doing exactly what he did. (b) Mr Levine didn't break any encryption scheme, didn't forge a digital signature, didn't find any subtle bug in the protocol and didn't do any of the kind of things which we in the cryptographic security community are concerned about. My experience with Citicorp and other organisations have led me to bring up a few points which I'll just read and perhaps we'll have a chance to talk about it during the discussion. The first is three or four observations about what is happening in the real world. I have one which I think is particularly relevant, and a couple of challenges which actually follow from these kind of observations. Out of the trenches, the attitude which I found in many organisations dealing directly or indirectly with issues of security, the question will come up: has it happened to us before, has it happened to our competitors before, is this particular system secure, how much is it going to cost and what is the ROI (return on investment) for that particular cost. As far as academic challenges, many, but I choose to name a few, security is not all or nothing, semi-formal approaches can get us quite a significant way along the path, education, tools for large complex systems, and finally, given the state of the financial world as I saw it, find a reliable place to deposit the research grant.

Mark Lomas: Can I start by asking the Chairman a question, which is was I invited here as an academic or as commercial representative?

Bruce Christianson: The answer is yes.

Mark Lomas: I should explain, I've been working for Goldman Sachs for the last year but as Stewart said in his introduction I spent a long time here in Cambridge and so my views on security may differ from some of my colleagues, my current colleagues. I would hope they differ less than from some of my previous colleagues. I haven't prepared a long talk so we can make up for Rafi's long talk. I'd like to essentially start by answering a question that Joan posed the other day. She said is the work that she's working on useful and my answer to that is to say that if somebody could come to us with a very good way of describing our security policy then we want to buy it because we have no way of determining what our current security policy is. It's too big, too complicated, it's administered by too many people, it's not properly understood and we've got severe problems in managing it. As an example of the sort of problem which we have we have statutory audits, we also have informal audits that we pay for because we want to make sure that we exceed the regulatory requirements placed on us, and we regularly get asked questions by both types of auditor, they say, please tell me everybody who can do this, where this could be some arbitrary complex function and we can't just give them a quick answer. If there's a particular system that they want audited thoroughly we would have to go around all of the different components and audit those, we would have to visit system managers scattered all round the world, we'd have to look at who had access to those systems, who could have changed some of the access control restrictions in some of the components that make up this large system and so honest answer to our auditor is we don't know. Fortunately, we don't think that we're losing too much money as a result of that but there's always the potential and part of my job is to estimate the risk and if I have such incomplete information I can't answer questions about risks so essentially what we would like are tools to help us manage things rather than people to come along and give us yet another protocol because if you invent some new application you say what the financial industry needs is this and we've got all this wonderful elaborate proof for it, the likely response is we don't even know how to manage what we've already got and you're asking us to complicate the systems that we've got by adding yet another component so essentially what I'm saying is keep things simple, make them easy to manage, make my life easy.

Joan Feigenbaum: So I guess I'd like to support Virgil's accusation that I'm an academic because I didn't actually say is the work I'm doing useful I said is it a good idea. You interpreted that — I think that answers your question of whether you're an academic.

Mark Lomas: Can I answer both of those questions in the positive, yes it's a good idea and yes it's useful.

John Warne: I'd like to take a viewpoint here that reflects more the view of industry to academia and I think there's an urgent need for academia and industry to work together and to do so to close a gap. Now if I naively define what academic practice is, it's really the pursuit of knowledge and research into specific problem spaces, abstractions and formalism, and commercial practice, believe it or not, does build upon academic results or practice to yield real systems. Now if you look at that we say, well how much of this research is actually brought into industry or into commercial practice and it could well be as small as you cited, but it's definitely a trend towards pulling through academic research into industry. But there's a problem if we look at this issue of delegation that was raised today. There are systems out there which have actually built the delegation in them and there are new systems being built with the notion of delegation. But we've more or less spelt out here that we've got some doubts about what delegation means so the question is what is industry actually building at the moment and maybe we need to close that gap by making them more aware of some of the questions that they need to answer or ask the academic community all the time. In other words this should be an on-going process. Not just build a system, good result, and wait for something bad to happen, it should be build the system and make observations about what academia are going to say about what might happen that's going to be bad. I think this is quite an interesting thing. Now from our point of view, and I talk about Nortel, the current focus is on the Internet and the Internet is a large confederation of disjoint entities. Now this leads to a number of observations. There's no global infrastructure and there's no single policy or mechanism. I feel it's very important to observe this because I think what we're trying to point to is the provision of dependable systems of which security is one attribute but not the total number. I would say that a system that behaves according to an authoritative specification is a dependable system providing you understand what you mean by authoritative. We won't go into that definition either, but what we're talking about is reliability, availability, safety tied into security, and how these things all come together in a real system, this is the important point I wish to make. There is no global infrastructure or mechanism, so the best hope for security, and I would just like to summarise two points, is local security to enforce local policies or mechanisms, that's extremely important, and if I take an object oriented view where I say an object is an encapsulation of state and the operations you can perform upon that state that could be a network, a virtual network or anything, but an object is an object. But an object has the ability to portray different views of itself and this access control issue that we've been talking about today, objects could have a single interface with refined interface types and project those different types to represent different access control policies on its state. And I think that's an issue that is sometimes overlooked with object orientation. OK that's the second, and the last point, I think the best hope for security is security to suit the purpose, a general toolkit for technology not a global platform for technology, and if we look at a toolkit we'd want the separation of concerns but we'd want them integrated, confidentiality, integrity, authentication, non-repudiation. These are concerns that need to be considered as part of the evaluation of the system design and then we need to say, well what are those things we are going to bring together in the particular specific instance of a platform that we're designing for a specific purpose. And I think the same rules apply to the remaining attributes of dependability and thus it's integration that's the issue. And the final point, I just wanted to raise as a controversial one that industry cannot ignore law and regulations and we at Nortel have to deal with lawful interception but I don't want to discuss that issue here today.

Bill Harbison: It seems that what I've heard says that what industry is looking for is some mechanism to evaluate and manage risk, however that is specified, they would like to have a hand in specifying their risk parameters and it could vary from company to company, I wouldn't say so much as day to day but it might vary from application to application within an organisation but what you're saying is you don't have a toolset which at the moment enables you to tackle the problem of evaluating where your risk is.

John Warne: Yes, also going back to Bruce's note of transparency, I don't wish industrial members to have this notion of transparency when they're doing this evaluation. They have to be expert in security, they don't want to actually research security, but they need to know how to apply security in a way that satisfies business requirements, and you can't do that by ignoring what security means. That's important to mention also.

Virgil Gligor: But one of the things that I noticed in my interactions with industry is that a lot of these outfits such as banks, equipment firms and so on don't have a very clear view of what their threat is. In other words they have some regulations that they have to follow because regulatory agencies tell them something about audit and they are very scrupulous as Mark pointed out about that but they have no clear definition of what the threat is and what I'm trying to say is that if you don't have a clear definition of the threat you certainly won't have a clear definition of the policy because a security policy or policies exist only because there is a threat and if you have no threat you have no policy whatsoever. So I'm not even talking about risk management, you can only do risk management if there is a threat and if there is a policy in place and you consider what's left being the risk, and until and unless they have a very clear definition of the threat they won't be able to have a clear definition of what their security policies are. Contrary to that in the academia we have very clear definitions nowadays of what the threat is, they may not be realistic, they may be overblown, they may be paranoid, but at least we've learned that to design anything we have to state explicitly what the threat is so I find that this is a very big difference between commercial concerns of security and academic concerns and this is one of the things we can make the commercial world very much aware of, the necessity to let them evaluate or list the threats and the policies that work.

Bill Harbison: Can I just follow on from that because I've worked on both sides of the fence so to speak but in the reverse order to most of the people here because I started out in industry and sort of drifted down into academia here, but my observation is that when I looked and started in security and started in research all of the security models that were being pushed and proposed assumed that everybody on the inside was OK and everybody on the outside wasn't, and yet all of the commercial results in risk analysis that one sees is it's almost the reverse of that. It doesn't matter what people on the outside do because the real threat does come from the people on the inside and Rafi, I don't know how much you can go into the CitiBank affair, but the reason they didn't need to crack protocols and other issues wasn't because of an external attack in the way that we normally model it in the security community.

Raphel Yahalom: Let me make two comments, one to Bill's comment and the other to Virgil's, and reflect again back on what is now called within CitiBank the St Pete affair after St Petersburg which is the location of the guy. I got involved right after that event so I have to thank every morning Mr Levine for managing to do a deal. Who said crime doesn't pay. But two observations. One, striking things that happened in the bank right after the event, right after they were beginning to realise what actually happened and without disclosing too many confidential information I may say that to this day, by the way the guy is only been extradited to the States about a month ago after spending a year in the UK at Her Majesty's pleasure, so not all the facts are still known to this day, not in a clear cut way, but two things which are interesting as an academic coming into that environment. The first thing once it happened the first reaction of the bank was to look into their own employees as the main suspects, so all the people I spoke to were very defensive and they've been through having a light shined in their face because they were the obvious, a lot of the assumptions were that there was at least an insider involved to some extent, so the fact that they don't look at their own internal employees is not accurate, it's sometimes quite extremely in the opposite direction. The other thing which was interesting, again as an academic, that led to Virgil's point, is that much of the effort after the event was not spent on security experts, internal or external or looking into how can we look at the vulnerabilities and close the holes, but much more, I would say much more resources were spent, resources in terms of manpower, money and so on, on getting public relations experts. That's the real threat model. What would the market say, what would the customers say, what kind of phone calls would they get. They did by the way a very good job, I was very impressed. The Wall Street Journal said everything about the incident was under control, at no point was clients' money under risk which again was very good public relations. And why is that the case? Because if you look at the high level management of such organisation, it's changing a bit but very slowly, you don't see security as being very high as far as the strategic high level management concern because people traditionally tend to associate with that with the technological side of the business as opposed to the real hard core mainstream type of financial organisation concerns. Which brings me to the point that part of the contribution which communities like ours can make is to provide tools which would allow to bridge the gap which is now closing but is still very obviously there, namely how can we say something about security which would make sense at the CEO level, the strategic level, in terms of things risk management, you know as good as it can be but making any sort of statement which would allow us to determine what is the value being provided, what is the risk which you are exposed to if you do or you don't do that particular step, I think that's a very significant contribution which needs to be made.

Bill Harbison: Well perhaps as Mark said, we might discuss policy more than we discuss protocols a little more.

Joan Feigenbaum: I have a question, why do you think security researchers would have anything to tell you or to tell Goldman about what its policies should be. I have exactly the opposite which is the last thing you want me to do is tell you your policies.

Mark Lomas: No, no I'm not asking you to tell me the policies, we've got several departments whose sold purpose is to set policy ...

Joan Feigenbaum: That's your problem!

Bruce Christianson: Rather, allow them to encode what they find out about policy.

Mark Lomas: Rafi's point was a very good one. We have various different departments because they're concerned with different aspects of the policy. For instance, we have a compliance department that tries to make sure that everything we do is legal and aboveboard and that we comply with all the regulatory requirements and they will draft parts of our security policy that say if you are going to put a system in place it shall have the following properties in order that we comply with all the laws and regulations. We have a management controls department, and they draft lots of regulations which are more concerned with protecting us from ourselves, they'll say how to do a division of responsibility in order to make sure there isn't one person who can issue a funds transfer and they will give us a set of rules. There's my department and we are more on the technological side but we will look at what we think are reasonable risks. In some cases it's just a matter of working out whether we comply with the set of requirements we got from the other two groups. In a lot of cases we draw attention to things that could be checked but it hadn't necessarily occurred to them and so it's a partnership, but part of the problem is, and this goes back to Rafi's point, how do you explain to people who have different backgrounds what the policy is. If you ask somebody in my department about something, what it is that enforces a particular rule, they will start talking about computer systems and networks and network routers and firewalls and if you gave any of those documents to our compliance department they'd say what on earth is this, half of our department have got legal training, they know nothing about technology other than as a means to an end and I would like them to be able to express rules in a way that our group can understand and also to be able to send back a set of rules that we believe we're enforcing and say does that actually meet your objectives.

Joan Feigenbaum: OK so there's translation between different native languages.

Mark Lomas: That's part of it.

Joan Feigenbaum: That's part of it but it sounds like you're saying there's something else. So our group's been working a lot on, you have a policy, you have something you want to do, you can plug it into our system and we'll tell you does that request comply with the policy, but that's sort of like a model checking approach. It's a limited notion of proof and compliance and all that stuff. Now it sounds like you might want to do something more general than that. You might want to say tell me all the things you could do or tell me all of the keys that might be able to sign off on this.

Mark Lomas: Well I can tell you exactly the questions that we want to answer. I want to say, tell me everybody who could possibly read the value in that account. Tell me everybody who can transfer money from this range of accounts. Give me the full set of accesses that Bill Harbison has.

Joan Feigenbaum: OK, so there is a problem with what you're asking. There's the general principle that I've heard from many people in security, most recently Rivest in that CRL paper, which is that you want to set up your security system and your signature and credential verification system so that it puts the burden on the requester to come up with all necessary credentials. That's what makes the problem computationally tractable we hope. It's still not all that tractable but it's tractable in the theoretical sense of tractable. Sounds like you want to eliminate that, you want to say, who has, or like how could this ever happen, according to some set of credentials.

Carl Ellison: I don't think he wants to eliminate that at all. What I believe is that he has a problem that you and I are not addressing.

Joan Feigenbaum: Right.

Carl Ellison: There is a database problem in a sense which is what you're trying to address.

Mark Lomas: Yes, I think so.

Carl Ellison: Just as there really are directory problems even though we say names are stupid things to use in the security protocols, people do need directories of names, and they need to make various inquiries in those directories. Those problems exist even though we've declared they're wrong for security protocols.

Joan Feigenbaum: Right so I think that maybe Carl just said it, maybe I think you've got as much a distributed database problem as you do a security problem.

Bill Harbison: Is there also not another problem which is a topology problem. You want to know the routes by which certain actions maybe connected and that may not be intuitively obvious initially.

Carl Ellison: Well, a graph theory problem in addition to a database problem.

Raphael Yahalom: Wait, I see a whole host of problems which you can map, in my mind I have it mapped from application to areas of computer science and other disciplines and I'll give you a couple of examples. However, just to clarify my point which may have been misunderstood Joan, I didn't imply that this is what I suggest computer scientists and researchers would necessarily address because knowing many of my mathematical friends they wouldn't care less and I don't blame them, but what I'm saying is that there is an opportunity which I think academic research is in a unique position to address and that's not only a database problem, or a certification problem although it's certainly part of it, when I look at some of the things like a funds transfer protocol that crosses end to end any financial organisation that's to me a very complex protocol without components but if you analyse that in exactly the same tools you may or may not use or we have or don't have as far as analysing the simple Alice and Bob type of scenario, even if you just are making small connected steps to say something about the protocol because you are already much further than what the current state of the art is is in the real world. Now what kind of expertise is required to do that kind of analysis? Well there are no magic answers but certainly I think the academic community or people related to that world are in a much better position than anybody else to do that necessary type of work, and that's not database problem but I would say quite close to the mainstream security analysis.

Stewart Lee: To what extent, when you say the academic community should be working in given area, is that because it's not under pressure to solve some specific problem as contrasted to the commercial community that really only looks at these when guys from St Petersburg break in.

Raphael Yahalom: Again I can only speak from my experience, but you know one of the things that is striking about the real word security and non-security experts in these organisations, is that they have much more pressing things to worry about than worrying about the more high level academic perspective, in particular the cash management system that Mr Levine broke into at CitiBank transfers in one day and that was before the merger last week which made CitiBank and Travellers the biggest financial organisation in the world now, at the time he broke into that cash management system the volume of dollars which actually flowed through that system in a day is four hundred billion. It's something like one fifth of the US's economy goes through the cash management back-end which credits one account and debits another. 90% of this four hundred billion are fully automated end to end, end to end meaning a bank in India transferring any amount which could amount to millions, to a bank in China, it goes all the way end to end without any human interference regardless of the amount being transferred. The only 10% put on somebody's workstation are not because it's an unusual amount or an unusual transaction, it's because somebody missed out the name of a Chinese bank somewhere and some low level operator had to correct it.

Dieter Gollman: I have a question about tying in within companies, triggered by Mark's remark that it is a problem that one part of the company are the legal people, the other part are the network experts and each group has its own private language and putting them in the same room keeps the separation intact. What's going on in companies in general in this area?

John Warne: Nortel's point of view of what they do is that they establish people like myself which in a sense take on a conscience for the company whereby I'm advised to find out what the problems are in the company in terms of real communication problems and things of that sort and then take a strategic position to try to get those things rectified if they can be, but it requires co-operation. The real question here is how much co-operation can you get in organisations and then when you get that co-operation what do you do to make it last. Don't go and given them a big problem over and above the big problem they've got, give them some tools and methods that are going to help them along, maybe even add value to what they've got and give them some simplified approaches.

Audience: That's what our company sort of establishes.

Raphael Yahalom: Let me touch on Stew's comment about to what extent the insiders within an financial organisation can take on the role of doing that kind of analysis which we are now pointing at the academic community, and I think it related to Dieter's point. Two issues, first, as I said the four hundred billion that moved through the systems are their responsibility. The liability I know ties into the discussion of this morning, but these people have real day to day transactions and concerns, most of these organisations, to take care of which are much more down to earth than some of the bigger picture issues we are bringing up. The second issue is again what type of people are available in this kind of organisation, what is their expertise to conduct the type of analysis that I'm proposing is necessary. Often, again for traditional reasons, you don't have the right set of skills to do the kind of things we are advocating, namely be able to look at big pictures, have the technical know-how, have quite extensive security cryptographic background but being able to map it into some sort of a business or financial model, that's something which most people in most financial organisations don't have the background.

John Warne: Can I come to that point, this notion of closing the gap. I'll ask you what sort of formula, do you think we could have to close the gap? And if you don't have an immediate answer to that I think we could point to something like Stewart Lee's lab. Here it's a multi-disciplinary lab which has been established by Roger in the first instance, it needs to be extended, but it gives an opportunity for industrial people to come and work in that lab on real problems that they understand with real academics, and I think if this kind of thing could become pervasive this would actually go a long way, providing there was a technology transfer back to the companies, and that in principle is established by virtue of having the industrial fellows, but ideally what you really want is the lab of Stewart Lee connected to the lab of company X and the technology transfer to be filtering through all the time.

Stewart Lee: I would be less than human if I didn't point out that we are open for business and are prepared to talk to other people.

Virgil Gligor: It's interesting that you mention this bridging the gap problem, we have to be careful not to bridge a much needed gap for the following reason. It turns out that there are relatively few areas of common interest surprisingly enough, I mean if you look at the statistics that, for example, Bob Courtney came up with in terms of security, he showed that about 65% of all the integrity problems which are of interest to the industry and business were actually fire, water, earthquakes, power surges, and the commercial world speaks in real money in order to protect against those events. Secondly, he pointed out what Bill Harbison mentioned namely that the rest after two percent which represented outsiders attacks were actually insiders. It's not our interest in the academia to deal with natural events nor do we deal with the very important problem to them which is physical security. I mean physical security is really the number one concern in the commercial world. We know very little about it and from my experiences at IBM I know that they're very concerned with physical security of PCs and have extremely sophisticated research in PC physical security. Remember that 4755 and the other boxes were sort of the commercial version of what they did in the late 60s and mid 70s for physical security of the cryptor in computers. So basically physical security is not something that we as a community are particularly interested in and that is sort of part of that gap that should not necessarily be bridged. So what we have to do first is identify the much needed gaps that will remain that we have no interest in and industry has an interest in and the other way round. Secondly, we have to identify areas of common interest and in those areas of common interest do we actually want to bridge whatever existing gaps may be, if there is a reasonable payoff for both and if you follow the money in terms of where we in the academia spend money or make money and where industry make money, it turns out that there are very few areas of common interest, surprisingly enough. We actually "make money", or use money to do fundamental research, research which the commercial world is not necessarily interested in for today's problem even tomorrow's problem, if tomorrow's problem is in the next two years. So we continue to do that — to do fundamental research in those areas that are likely to be of interest to the commercial world. On the other hand it's the commercial world's responsibility to tell us what are their fundamental problems, and even if they are in the span of five years or two years or one year or whatever, unless there is this marriage of interest the gap will continue.

John Warne: I'm certainly not suggesting otherwise. I'm not suggesting in this plea to close the gap that industry should constrain academia in what it is actually interested in and pursuing. What I'm saying is there needs to be an observation by both parties and unless you bring these parties together those observations would never be made.

Virgil Gligor: We don't know each others interests it seems, we actually seldom know each others interests and sometimes when we find what each others interests are we most often part ways, but on the other hand I think that there is one area of expertise that would be transferred very profitably, something that Rafi mentioned in passing, is methods and tools, analysis methods and tools. Why is that the case? It turns out that the commercial world sells function. Seldomly it sells quality to the dismay of the Japanese software industry. It turns out that if you sell function without quality then you are not interested in a wide variety of analysis methods and tools. It turns out that that's when we come in. We are interested always in analysis methods and tools and hopefully if we can make the industry or commercial world interested in analysis methods and tools then we have a common interest. So far I find that we have a very hard time just like the government did in the past convincing the industry that they need analysis methods and tools for security in general and the reason once again why that's the case is because they sell well — in the absence of regulatory laws generally. So what I think in the next ten to twenty years, what's going to happen is we'll be able to sell more analysis methods and tools because they'll be more on the box about negligence and liability and so on which would essentially be covered by sufficient analysis methods and use of tools. I think that there is a future of common interest because of the new laws coming up.

John Warne: Time to product is a big issue and essentially there are a lot of companies who would rather put a product in the market place and worry about those problems afterwards rather than suggesting that they should close their factory because they're not going to be able to compete with their competitors.

Virgil Gligor: I feel the Japanese lost in the 1980s because they never tried to put products out to be fixed by somebody else. Quality, they paid a very heavy price, a very heavy price by being late.

Peter Landrock: I think there's a limit as to how useful the academic tools for analysis, threat analysis, and what have you, is helpful. I think the problem is entirely different. If I can compare to a completely different part of the real world, "I started off building very advanced turbo engines but I couldn't sell that so now I'm a car dealer." All academics are good at something, one is very good at building ABS brakes, another one at building very excellent engines as I said. We all have our areas but we don't really understand how to build a car, but that's what the industry needs, so we must somehow refrain from insisting on being experts in some little area. We are being specialists and some of you know my favourite definition of a specialist is somebody who knows when to call an expert.

An expert is somebody who knows infinitely much about infinitely little and that's really the problem of most academics, but they think that this world is so grand that it can be useful for a number of things that may or may not be true. And the final remark that I want to say is that in a number of situations it's wrong to start with the threats because they are more or less self evident but I want to point out that there is a great difference between building a secure closed system in a large company and building secure electronic commerce with completely different entities because for instance the law will typically require that when you trade you must provide a handwritten signature otherwise it's not legally committing and therefore we have to provide digital signatures but the threat analysis is not trivial but it is self evident. When we developed the Bolero model for trading in electronic bills of lading there were also companies being responsible for the threat analysis and nothing came out of it that wasn't evident.

At this point the discussion was adjourned for a break.

Michael Roe: Talking about analysis tools and Carl Ellison's paper in the proceedings earlier: There is an interesting split sometimes, not always, between academia and industry where industry has a definite interest in things not being understood, where your entire business model is predicated on your customers being misinformed and not actually understanding how the product works. In these cases the coming together of academia and industry is very hard because academia is there to act in the public interest on behalf of these small customers and to say "The Emperor has no clothes!" There is a similar case where both the vendors and the government had a deliberate interest in misleading the rest of the public, and academia's job of actually understanding what the protocol really did was not one that was likely to be appreciated by any of the people with the money to sponsor doing it.

Dieter Gollman: I don't think it's always madness on the side of a company. Last summer when I was still an academic, a student came to me with a protocol which claimed to be an authentication protocol where the client at the workstation goes to some management server and gets the management server to send the key which identifies the user to some other server which eventually identifies the user. The student showed me the protocol and said it doesn't look right and I had a look at it and said yes after one run of the protocol this server can impersonate the user with respect to any other server on the systems, so I would be very reluctant to call it an authentication protocol. And just for information it had a four letter acronym starting with NT. I have seen this protocol since and it has been changed but these people had no clue what authentication protocols were about, there was no malice at all involved.

Bruce Christianson: I think the other problem which security has that's distinct from other areas where computer science and industry intersect, or fail to intersect, is that we've got to learn to be a lot more devious about the benefit that we are going to bring. We're currently in the position of the Californian sect, who when somebody asked why he should give them money, what is it that you achieve by what you do, replied that the previous year by the power of prayer alone they'd averted five major earthquakes. I think security is in a similar position, we're saying that we're worth all this money because of all the bad things that don't happen.

Joan Feigenbaum: This is why we can't have a good demo.

Bruce Christianson: Yes, a good demo would be self-defeating.

Virgil Gligor: We had one, Mr Levine did it.

Bruce Christianson: Well, quite, a good demo would be extremely expensive. So I think we have to come up with some cover story about what the benefit of security is other than making systems less available to legitimate users or all current perceived benefits of security.

Mark Lomas: Well a cynic might suggest that one of the reasons that my department is hired is in order that my employer can show due diligence, and so if I ever were to say I'm satisfied with everything then they would start getting really worried because they want to be able to tell the auditors that things are improving year by year, so, keep giving us things we can tell the auditor.

Joan Feigenbaum: So how about the following as a direction for basic research. A risk management oriented and quantified version of definitions and theorems and proofs that we have in the security world, so I think this has been said before in this workshop, to be able to say something like, we've analysed our system, we've done asset identification, this is what we're trying to protect and this is what it's worth to us. Now what someone who really understands computer security should be able to deliver is a cost effective way of protecting this asset. In particular not something that can only be used in a way that costs more than the thing that you're trying to protect or that requires an incredible amount of co-operation from the entire world to comply with, or something that's very unrealistic. I guess the other part of that is to say, not to promise some kind of ridiculously rosy sounding level of security, because I think any good businessman will react to that by saying I don't want that, that obviously is the expensive version, I'm not trying to protect, you know ....

Raphael Yahalom: Don't bet on that, I mean this industry of security consultants which has always been there has blossomed over the last two years, partly because of the exposure and event like Mr Levine and like the Internet and what I see around me in terms of including big names of security consulting firms is that financial organisations spend big amounts of money sometimes for the issues of due diligence, sometimes because they really believe that they need to get the benefit. They spend six, seven figures sums and more and at the end of the day they get something very thick and impressive, in fact bound like the Bible except that they contain many more miracles.

The industry exists, there is input coming in to these organisations, the question is who is it coming from, what is their value, are the people, the decision-makers within the organisation and consequently the market, the users, us, the customers, are we able to distinguish or will we be able to distinguish, perhaps with trust and intermediaries, between a service provider which does a good job and one which doesn't do a good job, I mean that's nothing new, but I think these kind of things will be based on market perception as opposed to regulation or law.

Joan Feigenbaum: OK let me respond to that. I don't think you can say they spend six or seven figures on this therefore it was too expensive. The budget of that organisation is the in hundreds of billions, that's not a very big expense.

Raphael Yahalom: I didn't mean about a big or small investment, I meant in respect to what are they getting back.

Joan Feigenbaum: Yes, exactly. No, no, that's not what I meant, not in terms of what are you getting back, what I meant was in terms of what you are trying to protect. I think you want an answer from your security consultant about what mechanisms you should use or what policy you should use, or the whole shebang, the whole end to end security system, you want an answer that is commensurate with and proportional to, and reasonable with respect to what it is you're trying to protect. Our whole theory, all of our terminology, all of our definitions, is not well suited to do that, all the definitions of one way function, of zero knowledge, of signature, of, you know, lots of stuff is not amenable to. I have one million dollar assets to protect I want the right protocol, I have one billion dollar assets to protect I want the right protocol. You know, it's just not set up that way and that kind of risk management framework layered on or intertwined with our complexity theoretic framework, if I may Peter, would be very useful.

Raphael Yahalom: That's exactly what I mean.

Dieter Gollman: In continuation of Rafi's comment, the banks, companies, go to those people who can speak a language they understand and academics don't try very hard to speak that language and in terms of gaps there is a language gap and I think we should, as academics, instead of pretending to be rather an academic, try to do something about it. That has nothing to do with the gap in research interests. Back at Royal Holloway, where I was before joining Microsoft, we were teaching this Masters course on information security, most of the people teaching only the mathematics and for the fun they did things about sequences and perfect sequences but they wouldn't touch the topics on the MSc. where we try to tell people about real security. And maybe related comments

- academia is less about the pursuit of knowledge, it's about the pursuit of publications, it's about the pursuit of short-term publications which makes all this real industrial approach really nasty because it takes time to understand this problem. Larry was speaking about how long it took him to understand TLS which is a fairly simple thing compared to the organisations of Goldman Sachs' size, and the things they're doing and so it's not attractive for academics in their current reward model.

Mark Lomas: I think that's partly a fault of the people that specify things like TLS rather than something that academics ...

Stewart Lee: You want it more complicated?

Mark Lomas: No, I'd like to be able to read a specification on one page of paper because I can then think about it.

Eric Barber: Some attacks are lawful in the sense that companies like Nortel are obliged to allow government to intercept, I wonder if academics are considering a theory of trapdoors or if the standard Alice and Bob model is sufficient to allow governments to intercept and to look at transactions without the participants to the transaction, without Alice and Bob actually knowing about the interception.

Raphael Yahalom: I think every respectable researcher in crypto has at least one paper which expresses one's proposal for a new scheme which would allow or disallow some sort of an escrow or key recovery, so certainly work is being done now along these lines.

Carl Ellison: It's also work funded by ARPA that I know about, to study this problem in general.

Virgil Gligor: That's Hillarie Orman's project. But also she funds something which is quite the opposite. They are looking at situations in which the mechanisms are way too strong, way too inefficient for the threat model that they're addressing. In other words if you always make the worst assumptions under the world about the system, even though they're realistic assumptions you're going to end up building systems that are unrealistically, I mean performance that are realistically poor, so there is an interest in that as well, not just the man in the middle type of assumption or the insider assumption.

Carl Ellison: As someone who did research in key recovery, it bothers me when we get excited by the academic challenge and pour work into that without at the same time making a strong statement about the morality or immorality of the government claim that they have a right to this access.

Raphael Yahalom: Well don't worry, academics don't keep quiet about that, in fact key recovery is a good example of the relationship between the academics and the real world so to speak. I happened to attend a panel last week at MIT, I mean I was in the audience but on the panel were Whit Diffie, Ron Rivest and Barry Smith I think from the FBI.

One of the points that were raised was about this document which I think most of the people in the room have read, written by Ron Rivest and Matt Blaze and Ross Anderson. Quite a few well know cryptographers and computer security people wrote a very eloquent document which I personally disagree with, but basically a very eloquent document. It came up in discussion, they asked for the formal US government comment to that document, and essentially what the FBI said essentially was that they read it "took it into consideration", that's it.

John Warne: Just one last point. I belong to a number of standards organisations and we're beginning to see government officials actually make statements about why it is necessary to relax the rules about cryptography in order that commercial organisations can do their business without fear of losing secrets, and so I think there's a morality that's developing even within government circles. How it gets resolved, well that's ...

Optimistic Trust with Realistic *e*Nvestigators
(Position Paper)

Raphael Yahalom

MIT - Sloan School of Management
50 Memorial Drive
Cambridge, MA, 02139, USA
yahalom@mit.edu

Abstract. We present a new notion, *Optimistic Trust*, which may serve as a useful design principle for a variety of practical application protocols. Specific arbitrator entities, *eNvestigators*, on which *Optimistic Trust* relies, are introduced. A new practical electronic election protocol based on the new approach is outlined.

1 Introduction

The notion of trust can be viewed as an assumption by some entity that another entity will behave in a certain way in particular circumstances (e.g. [16]).

In most electronic protocols, as in most real-world interactions, the attainment of the desired goals of an interaction is dependent on players behaving consistently with certain trust assumptions (e.g. [3,9]).

The type and scope of necessary trust assumptions may be diverse and subtle (e.g. [10,16,2]).

Often, there is a trade-off between the level of trust assumptions and the performance costs of a designed protocol. The fewer (or weaker) the trust assumptions the higher the performance cost and the complexity of a protocol designed to attain given goals.

Obviously trust may be violated. That is, a trusted party may behave inconsistently with expectations. Clearly the likelihood of such violations depends on environment, circumstances, and application characteristics.

An entity may betray the trust of others for different reasons which may be categorized to two main categories:

1. *Opportunistic behavior* - the entity may gain some benefit by violating the trust in particular circumstances.
2. *Honest-Mistake behavior* - the entity may behave erroneously with no intention to attain any goal by such deviation [1].

[1] An entity's *intentions* include the intentions of external principals (such a entity designers) who may have influence on the entity's behavior.

In real interactions, depending on the protocol specific features, an entity that violated trust for opportunistic reasons may often attempt and plead *"honest-mistake"*, if detected and accused. Depending on the protocol design features, evidence supporting or contradicting such a claim may or may not be available to the accuser.

By *optimistic trust* we refer to trust which is conditional on the ability to detect corresponding violations and recover from them, after the fact.

In particular, as discussed below, optimistic-trust based protocols incorporate well-defined specification for circumstances in which optimistic trust assumptions are violated.

The premise underlying the approach presented in this position paper is that trust violations may normally expected to be relatively rare, however not impossible. Thus, rather than pay the extra complexity cost up front, protocols are designed so that every relevant trust violation will be detected. The extra performance cost will be associated with fall-back recovery phases of the protocol. Thus such price is expected to be paid only relatively infrequently - only when violations actually have occurred.

Consequently our premise is that protocols that are designed with such a principle are expected to be more practical and usable, without necessarily sacrificing security or exposing the participants to higher risk exposures.

2 *e*Nvestigators

Our notion of optimistic trust between entities captures an attitude of "I am willing to trust you in certain respects, knowing that if you violate the trust assumptions you will be detected and incriminated".

Such a notion is in a sense self enforcing: in principle, an entity is less likely to behave opportunistically the higher the perceived risk of loss.

The detection and prosecution is performed by *passive* entities, *e*Nvestigators, which join a protocol execution only after being invoked by one of the active protocol participants as a result of a detected exception (such an entity assuming that role is also referred to as an *e*Nformant). The *e*Nformant provides relevant evidence to the *e*Nvestigator as part of its invocation.

***e*Nvestigators:** *Passive entities designed to be invoked by one of the active participants of a protocol in the case of an exception and participate in an eNvestigation and recovery phase of a protocol, while ensuring protocol goals.*

An *e*Nvesitagation phase of a protocol aims to recover the correct state to the extent possible and establish the guilty trust violator, while ensuring that protocol goals are maintained.

Obviously an *e*Nvestigator may itself be corrupt and violate trust. Overall protocol and *e*Nvestigation phases design should reflect such assumptions.

Because trust violations are expected to be relatively rare the *e*Nvestigation phases are expected to be executed relatively infrequently. Thus performance gains may be achieved by designing the main phase of the protocol as efficiently as possible by relying on optimistic trust assumptions, while relying on the *e*Nvestigation phases to recover when these assumptions turn out not to have been justified. The extra performance cost thus seldom needs to be paid - only when trust violations do occur.

Note that an *e*Nvestigator can be regarded as a particular type of a conventional *arbitrator* entity (e.g. [15]). In particular, an *e*Nvestigator is considered as one of the protocol execution players. The specification is designed to exclude, if possible, the actual participation of the *e*Nvestigator in any given run.

The notion of an *Invisible Post Office* introduced by Silvio Micali in [12] can be considered as an example of an *e*Nvestigator. In the Certified E-Mail protocol presented in [12], *Invisible Post Office* entities participate, and guarantee a fair exchange, only if something goes wrong.

3 Election Protocols Perspective

We demonstrate aspects of our approach in the context of electronic elections protocols.

Similar principles apply in a wide variety of electronic commerce protocols and other application protocols.

Electronic election protocols have been the focus of a significant body of research in the last decade, though relatively few schemes have been implemented and used.

Electronic elections schemes represent an example of protocols in which both integrity and confidentiality goals are fundamental.

In general, the electronic elections schemes in the literature may be classified to three categories:

1. Protocols with no trust at a central authority, (e.g. [7]).
2. Protocols with much trust in a central authority, (e.g. [14]).
3. Protocols with limited trust in several central authorities, (e.g. [1,8]).

Whereas the first category is often considered impractical for performance reasons, the second category is often considered impractical for risk exposure reasons.

Thus most schemes that are considered as being pragmatic depend on a (usually small) number of servers, between which trust is shared in a variety of ways.

One such pragmatic scheme is the Fujioka, Okamoto, Ohta (FOO) protocol [8]. At least 3 variants of the FOO protocol have been proposed and implemented: at MIT [11] , at Princeton [6] , and at Washington University [5].

We provide an overview description of the FOO protocol. It assumes an environment of two servers, an *Administration* server (which we denote A) and a *Counter* server (C).

Let us assume that some list of n eligible voters V_i is reliably publicly available prior to the beginning of an election protocol [2].

In the notation we use below, $\{ \; msg \; \}_{K_{P+}}$ represent the public-key encryption of msg with P 's public key, and $\{ \; msg \; \}_{K_{Q-}}$ represents the concatenation of msg with a digital signature of a one-way hash function of msg with Q's private key.

We denote by $\mathcal{CO}_s(msg)$ the *bit-commitment* of msg with secret value s , [13], and by $\mathcal{BL}_r(msg)$ the *blinding* of msg with a factor r , [4]. A comma $''\;,\;''$ denotes concatenation between message components.

We assume that all the cryptographic primitives used are strong in the conventional sense, and that all keys and random values are of sufficiently high quality (e.g. sufficiently long).

The FOO protocol consists of the following steps:

First Phase:

1. V_i: creates her vote $ballot_i$
   generates a random s_i
   creates a bit-commitment $\mathcal{CO}_{s_i}(ballot_i)$
   generates a random r_i
   creates a blinded message $M_{1i} = \mathcal{BL}_{r_i}(\mathcal{CO}_{s_i}(ballot_i))$

2. $V_i \;\rightarrow\; A$: $V_i \;,\; \{ \; M_{1i} \; \}_{K_{V_i -}}$

3. A: verifies V_i eligibility and V_i's signature of M_{1i}

4. $A \;\rightarrow\; V_i$: $\{ \; M_{1i} \; \}_{K_{A-}}$

5. V_i: unblinds with r_i and verifies $\{ \; \mathcal{CO}_{s_i}(ballot_i) \; \}_{K_{A-}}$
   If verification fails, claims by publishing $\mathcal{CO}_{s_i}(ballot_i)$
   and A's response

End of First Phase:

6. A: publishes a "first-phase-voters" list of m entries
   such that each entry i is $[\; V_i \;,\; \{ \; M_{1i} \; \}_{K_{V_i -}} \;]$

7. *anyone*: verifies V_i eligibility and V_i's signature of M_{1i} for each V_i

[2] The generation and maintenance of such a list represents of course in itself many interesting protocol challenges, but such protocols are beyond the scope of our current discussion.

Second Phase:

8. $V_i \rightarrow C$: (via *anon channel*:) $\{ \mathcal{CO}_{s_i}(ballot_i) \}_{K_{A-}}$

9. C: verifies A's signature of $\mathcal{CO}_{s_i}(ballot_i)$
generates a sequence number j

End of Second Phase:

10. C : publishes a "second-phase-votes" list of p entries
such that each entry is $[\, j \,,\, \{ \mathcal{CO}_{s_i}(ballot_i) \}_{K_{A-}} \,]$

11. V_i: verifies $\{ \mathcal{CO}_{s_i}(ballot_i) \}_{K_{A-}}$ in published list

12. *anyone*: verifies A's signature of $\mathcal{CO}_{s_j}(ballot_j)$ for each entry
verifies $p = m$

Third Phase:

13. $V_i \rightarrow C$: (via *anon channel*:) $j \,, s_i$

14. C: obtains $ballot_i$ and verifies vote validity

End of Third Phase:

15. C: publishes a "final-votes" list of q entries and a results "*summary*"
each entry j is $[\, j \,,\, \{ \mathcal{CO}_{s_i}(ballot_i) \}_{K_{A-}} \,,\, s_i \,, ballot_i \,]$

16. V_i: verifies $ballot_i$ in published list

17. *anyone*: verifies $\mathcal{CO}_{s_j}(ballot_j)$ and $ballot_j$, for each entry
verifies consistency of results "*summary*"

As described in [8] and as demonstrated by the fact that it was relatively widely used as a basis for implementation prototypes, the FOO protocol is associated with various attractive elegant features, and may be considered as quite realistic as far as performance requirements are concerned.

However the FOO protocol is also associated with a few potential weaknesses related to trust violation circumstances.

We briefly discuss a few of these:

– Certain aspects of the protocol are underspecified. For example $\{ M_{1i} \}_{K_{V_i}-}$ (step 2) contains only a random string, and so A (having obtained such a value $\{ M_{1i} \}_{K_{V_i}-}$ in a different context) may be in a position to violate trust by submitting a vote to C on behalf of a voter V_i claiming falsely that she participated in the first round.
A's incentive to violate trust in that way may be significant as it may not necessarily be possible for any entity to prove his guilt.

- The verification failure claim in step 5 results in the disclosure of V_i's blinding factor r_i and so V_i may need to be allowed to participate again in a sequence of steps 1-4 in order to complete her voting session correctly and in a confidential manner.
  However, V_i may have lied in that claim, accusing A falsely and producing a bogus r'_i, while retaining the original correct $\mathcal{CO}_{s_i}(ballot_i)$, and thus possibly submitting multiple votes.
- A is in a position to violate trust by submitting a vote (steps 8-12) on behalf of all eligible voters who participated in phase 1 but for any reason did not complete phases 2 and 3.
- Every voter V_i is required to perform a variety of quite elaborate integrity verification steps unilaterally. Significantly, each voter also has to remain on-line for long periods of time spanning the whole election period (e.g. multiple election-related on-line sessions per user during a period of 2-3 days).
  In particular, such burden, in addition to being inconvenient, increases the possibility of voters failing to complete their voting process correctly, and thus providing A with additional opportunities to violate the trust in the sense of the previous point.

Aiming to design an effective pragmatic election scheme Mark Herschberg designed and implemented a FOO-variant (which we refer to as the Herschberg-FOO protocol), as part of his Master project at MIT LCS [11]. An implementation based on the Herschberg-FOO protocol will shortly be deployed for campus-wide elections at MIT.

The Herschberg-FOO protocol addresses some of the potential weaknesses of the FOO protocol and incorporates various effective implementation aspects. It incorporates an *Anonymity Server* as a method for implementing the *anonymous channel* required by the FOO protocol. The interaction of each voter with the elections protocol is significantly shortened, as the voter submits her vote (step 6) to the *Anonymity Server* together with the commitment key (step 13) - both encrypted with C's public key. She thus completes her involvement before the third phase in which C actually receives the votes.

Thus each voter's burden is reduced, as is the opportunity for A to exploit incompleted votes.

The Herschberg-FOO protocol relies on an *Commissioner* entity to oversee the correct execution of the protocol and deal with complaints. A *Commissioner* entity is related to the conventional arbitrator entities(e.g. [15]) in that, unlike an *e*Nvestigator, it is considered as an entity external to the protocol execution and whose interactions are not fully specified as part of the protocol.

The Herschberg-FOO protocol is also associated with a few potential trust violation weaknesses. For example:

- Due to pragmatic implementation environment constraints, each voter authenticity of the first phase voter's message is based on shared password between V_i and A. Clearly that allows A to vote on behalf of absent eligible voters, or to modify the submissions of participant voters without being detected or proven guilty.

- Certain aspects of the consistency control checks are under-specified. For example C may in some circumstances omit various valid votes (e.g. unfavorable ones) by claiming that they are invalid, without any entity necessarily being in a position to dispute that, or prove C's violations.
- Each voter V_i is still required to interact separately with two servers and thus may still be vulnerable to inconsistencies resulting from the correct completion of one step without the other.

We propose below a new protocol, inspired by the FOO and Herschberg-FOO protocols, which addresses various such trust-violations concerns by relying on our approach of optimistic trust and on the use of an *e*Nvestigator.

Such an approach enables us to reduce the potential risk exposures of the protocol while at the same time maintaining, and even improving, its attractive execution performance characteristics.

4 A New Elections Protocol

We present a general overview of the new protocol, omitting some of the details. A detailed description of the new protocol and its comprehensive analysis will be presented separately.

The protocol aims to attain conventional election protocol goals, similar for example to these in [8]. Thus the integrity of the vote aggregation, as well as the confidentiality of the votes of individual voters, need to be guaranteed.

The protocol incorporates a single *e*Nvestigator denoted E. Any message an entity sends to E is a result of an exception condition detected by the *e*Nformant. Such a message triggers an *e*Nvestigation protocol-phase and contains the relevant evidence by the *e*Nformant for the *e*Nvestigation initiation.

In the overview below, we omitted the detailed description of the supplied evidence, and of the specified *e*Nvestigation phases.

In our description, encrypted and signed message components include a *component identifier* denoted cid_m. Such cid_m contains header fields for a component m of a message, including such fields as message type, a unique global identifier of the elections, etc.

We specify the main phase of the new protocol. It is expected to normally be the only phase to executed, when no trust violations occurred. It consists of the following steps:

<u>First Phase:</u>

1. V_i: creates her vote $ballot_i$
generates a random r_{1i}
creates a blinded component $M_{1i} = \mathcal{BL}_{r_{1i}}(ballot_i)$
generates a component $M_{2i} = \{\ c.id_{2i}\ \}_{K_{V_i-}}$
generates a component $M_{3i} = \{\ c.id_{3i}\ ,\ r_{1i}\ \}_{K_{C+}}$
generates a component $M_{4i} = \{\ c.id_{4i}\ ,\ M_{1i}\ ,\ M_{3i}\ \}_{K_{V_i-}}$
generates a message $M_{5i} = \{\ c.id_{5i}\ ,\ V_i\ ,\ M_{2i}\ ,\ M_{4i}\ \}_{K_{A+}}$

2. $V_i \rightarrow A$: M_{5i}

3. A: verifies V_i eligibility and integrity of message
stores internally M_{4i}
generates $\{\ M_{1i}\ \}_{K_{A-}}$ by signing
(periodically) appends M_{3i} to a Web-posted
"confirmations" list and re-signs

3. C: (periodically) checks "confirmations" list posting and appends its signature

4. V_i: verifies that M_{3i} is posted on signed "confirmations" list
logs off.

<u>OR</u> $V_i \rightarrow E$: $\{\ \underline{evidence}_{V_i}\ \}_{K_{E+}}$

<u>End of First Phase:</u>

5. A: signs and Web-posts a randomly ordered list "voters"
such that each entry is $[\ V_j\ ,\ M_{2j}\ ]$
generates and signs a randomly ordered list "blind-votes"
such that each (signed) entry is $[\ \{\ \{\ M_{1f}\ \}_{K_{A-}}\ ,\ M_{3f}\ \}_{K_{A-}}]$

6. $A \rightarrow C$: $\{\ A\ ,\ \{\ c.id_{1A},$ "blind-votes" $\}_{K_{A-}}\ \}_{K_{C+}}$

7. C: checks consistency of "blind-votes" , "confirmations" and "voters" list unblinds each $\{\ M_{1i}\ \}_{K_{A-}}$ using r_{1i} and checks resulting $\{ballot_i\}_{K_{A-}}$ signs and Web-posts a randomly ordered list
"results" such that each (signed) entry is $[\ \{ballot_j\}_{K_{A-}}\ ]$

<u>OR</u> $C \rightarrow E$: $\{\ \underline{evidence}_C\ \}_{K_{E+}}$

8. **A:** checks consistency of "results" list and appends its signature

<u>OR</u> $A \rightarrow E$: $\{\ \underline{evidence}_A\ \}_{K_{E+}}$

Although we have not provided the specifications of the *e*Nvestigations in this position paper, we note that in some the *e*Nvestigator may draw the conclusions directly from the evidence provided by the *e*Nformant (for example an inconsistent message component, or set of components, signed by an accused entity), whereas in other *e*Nvestigations additional information needs to be obtained by the *e*Nvestigator with appropriate consequent message exchanges.

For example, in the above protocol, C may claim that a certain vote is unreadable following the unblinding operation. The *e*Nvestigation triggered by such a claim may require A to provide M_{4i} (incorporated by V_i in step 1 for that purpose) in order to prove A's innocence. Obviously the exchanges associated with such disclosure need to be designed so as to guarantee that none of the parties may maliciously gain information regarding the association between V_i and her vote.

The new protocol is designed with the assumption that any one of the three entities A, C, or E may violate the optimistic trust [3].

The specification of the various *e*Nvestigation phases, and a formal correctness analysis of the new protocol are omitted from this paper.

Given the relevant environment assumptions such an analysis leads to the following claims:

– If none of the elections servers violate trust, the election will complete with a signed result which correctly reflects *all* the votes of the correctly behaving eligible voters, and *only* these votes.
– If none of the elections servers violate trust, the vote of any correctly behaving voter will never be known to anyone.
– If any one of the election servers violate trust in any way - he will not be able to cause incorrect election results to be generated.
– If any one of the election servers violate trust in any way - he will not be able to discover the vote of any voter.
– If any one of the election servers violate trust in any way and causes the elections to be terminated, an incriminating proof against that server will be published.
– An *e*Nvestigation phase will be executed only if some trust violation has actually occurred.

5 Conclusions

The approach of optimistic trust with *e*Nvestigators design enables to improve the usability and efficiency of designed protocols, without sacrificing the security and risk exposure characteristics.

In the new protocol presented above security *and* performance were improved.

The design of such optimistic-trust protocols and the corresponding *e*Nvestigation phases may be quite subtle, and efficient proof and analysis techniques remain an important challenge currently being addressed.

[3] For cases in which multiple servers may may jointly violate trust, additional *e*Nvestigators would need to be incorporated in each *e*Nvestigation.

Exploring aspects of the approach in the context of other application protocols, such as electronic commerce ones, is also a very promising area of work.

References

1. J. Benaloh and M. Yung "Distributing the Power of a Government to Enhance the Privacy of Voters", *Proceedings of the 5th ACM Symposium on the Principles of Distributed Computing*, 1986, pp. 52-62.
2. M. Blaze, J. Feigenbaum, and J. Lacy " Decentralized Trust Management" *Proceedings of the 17th Symposium on Security and Privacy*, pp. 164-173, IEEE , 1996.
3. M. Burrows, M. Abadi, and R. Needham " A Logic of Authentication" , DEC System Research Center technical report 39, California, 1989. Also in *ACM Transactions on Computer Systems*, Vol 8, Num 1, pp. 18-36, 1990.
4. D. Chaum, "Security without Identification: Transaction Systems to Make Big Brother Obsolete", *Communication of the ACM*, Vol 28, No 10, pp. 1030-1044 Oct 1985.
5. L. Cranor and R. Cytron, "Design and Implementation of a Practical Security-Conscious Electronic Polling System" WUCS-96-02, Washington University, Department of Computer Science, St Louis, Jan 1996. *http://www.ccrc.wustl.edu/~lorracks/sensus/*
6. B. Davenport, A. Newberger, and J. Woodward "Creating A Secure Digital Voting Protocol for Campus Elections", *http://www.princeton.edu/~bpd/voting/paper.html*
7. R. Demillo, N. Lynch, and M. Merritt, "Cryptographic Protocols" , *Proceedings of the 14th Annual Symposium on the Theory of Computing*, 1982, pp.383-400.
8. A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large Scale Applications", *Advances in Cryptology - AUSCRYPT '92, Lecture Notes in Computer Science*, Vol 718, pp. 244-251, Springer-Verlag, 1992.
9. L. Gong , R. Needham, and R. Yahalom "Reasoning about Belief in Cryptographic Protocols", *Proceedings of IEEE Symposium on Security and Privacy*, May 1990, pp. 234-248.
10. W.S.Harbison, "Trusting in Computer Systems", University of Cambridge, Technical Report 437, 1997.
11. M. Herschberg, "Secure Electronic Voting over the World Wide Web", Master thesis, MIT - Laboratory of Computer Science, May 1997.
12. S. Micali, "Certified E-Mail with Invisible Post Offices", *Proceedings of RSA Conference*, San Francisco, 1997.
13. M. Naor, "Bit Commitment using Pseudo-Randomness", *Advances in Cryptology - Crypto '89, Lecture Notes in Computer Science*, Vol 435, pp. 128-136, Springer-Verlag, 1990.
14. K. Ohta, "An Electronic Voting Scheme Using a Single Administrator" , IEICE, Spring National Convention, A-295, 1988 (In Japanese).
15. B. Schneier, "Applied Cryptography", *2nd Edition, John Wiley & Sons*, 1996.
16. R. Yahalom, B. Klein, and T. Beth, "Trust-based Navigation in Distributed Systems", *Computing Systems Journal*, Vol. 7, Num. 1, 1994.

Optimistic Trust with Realistic Investigators
(Transcript of Discussion)

Raphael Yahalom

MIT

This talk focuses on some notions I have been thinking about in the context of electronic commerce protocols and in particular contractual relationships between parties in electronic commerce protocols. However I will talk about these notions and demonstrate them in a slightly different context, namely that of electronic election schemes and in particular a couple of supposedly practical electronic election schemes.

I will start by just saying a few words about these underlying notions of optimistic trust and investigators. Why optimistically trust? I want to spend one minute reviewing why do we actually trust an entity, what makes us believe an entity or expect an entity will behave according to some specifications. Well, there are a whole host of issues, many of them environment and application dependent but in particular there are two types of situation that we know may cause an entity to deviate from our expectation of trust. [Firstly] opportunistic behaviour of the entity or some other principal which has influence on the entity's behaviour, namely by deviating from my specification. As far as some cost benefit analysis the risk of being caught versus the gain expected to obtain by deviating from the specification makes it worth my while. The second type are just simple honest mistakes at various levels, either of the design or implementation or what have you. OK, so I think most of the deviation, most of the trust violation which happens in theory or in real life may to some extent fall under one of these two categories. Well the approach of optimistic trust is really based on a premise which basically says entities may violate our trust assumptions but in real life and in most expected situations entities are likely to behave honestly or behave according to their specification. So as a result we want to adopt some sort of an attitude which says we'll hope for the best but prepare for the worst, namely, we will design our protocols with the assumption that protocols do behave consistently with a specification, with our assumptions but have recovery mechanisms, sort of fall-back procedures which if the entities violate the trust assumptions we are going to fall back on these and still make sure our invariants, our guarantees are maintained. So the optimism here is based on the fact we are in most cases likely to gain as far as the trade-off between performance cost and trust assumptions, in other words we are going to gain some of the advantages of improved performance by assuming that the entities will behave according to the specification. However, we are not sacrificing ourselves or putting ourselves into a bigger risk exposure if something bad happens.

How do we actually achieve that? Well, we introduce this notion which I call investigators which again in itself is not something which we have not completely seen before but the way we're using these entities is slightly unique, I believe, and these investigators are passive entities, they don't normally participate in the normal execution of the protocol. However, they are part of the protocol specification in a very well defined way. In particular, by passive entities we mean that they are only involved in case one of the entities detects some sort of anomaly or exceptional behaviour. At that point an investigator, or more than one, are involved, they participate in the protocol execution and the definition of the protocol guarantees that the protocol with the investigators, under certain assumptions of course, will achieve the desired goals. Needless to say the investigators themselves are entities like any other entity and consequently each one of them can be dishonest and violate the trust for the normal reasons. OK, let's move on and demonstrate these points in the context of a couple of electronic election schemes, in particular propose a few [changes] to existing pragmatic electronic election schemes. We've all come across and most of us have thought about electronic commerce protocols. Much research has been done on these protocols with relatively few implementations of real systems.

An interesting experience I had with an electronic election scheme system was in Tel Aviv University a few years ago, a system that was designed to replace the student election procedures. Fully automated PCs, networks. As you know the one thing which it is not short of is cryptographers. I wasn't involved in that but there were people looking at all sides of the security aspects of the system. It had a nice interface, the voter came into the voting booth, chose four of the candidates, pressed enter, and the votes went to some sort of central authority and were tallied at the end of the day. At the end of the day when the election was completed, the organisers checked what happened with the results and they saw that four names had won the election. A closer look at the results showed that four winners actually received all the votes of all the voters, which again is possible but rather unlikely. A closer investigation of that system revealed the fact that there was a programming bug in the code and after the first vote of the first voter of the whole day some loop wasn't initialised so whoever you voted for the system acknowledged but somehow the four first recipients of the vote received all the votes recorded under their names. Of course when that was discovered at the end of the day, there was no way to recover the original votes of the voters — that was one of the characteristics of the system design — so the election had to be cancelled and re-run again a few weeks later, this time using pieces of paper.

Stewart Lee: Demonstrating the benefits of testing.

Reply: Yes, yes, indeed.

Now if you look at the literature at what is available and there are probably a few tens of electronic election protocols which have been at least proposed with different properties and assumptions, etc., I think they roughly fall into three categories.

One is the category which there is almost no, or no central authority at all. There is no trusted entity. The participants are paranoid and they don't trust any central authority, and they keep exchanging messages until they are all convinced of the correct result of the election. That's one rather extreme end of this paranoia spectrum, the other side of the spectrum is one of complete trust in some central authority which actually the participants trust, submit the votes to and that authority has the responsibility of handling the elections correctly. Something akin I guess to what is happening in Baghdad when Saddam Hussein wins something like 98% of the vote, 99% of the vote, in the elections there. Most of the other, slightly more realistic schemes, rely on a few, hopefully not too many servers and the trust is shared somehow between these servers, with various assumptions about how these servers interact. What are the assumptions about the number of dishonest servers, etc. etc.

So when it comes to implementing the real system, one is faced with a possibility of using one of dozens of different election schemes, the question is if you want to implement one, which one do you choose. Some of them even include unusual features, more exotic features. For example, there is one election scheme which allows the voters to change their mind after the fact and change their votes, which when I read that paper I was wondering how Binjamin Netanyanhu, the Prime Minister, what would he be doing these days if that option was available to Israeli voters in our election.

So I want to refer in particular to one proposal which was proposed about five years ago by Fujoka, Okamota and Ota. I refer to it as the FOO protocol. It is a protocol which was published in the reference list in the lecture notes of computer science in about 1992. This protocol has nothing really unusual about it, nothing very exotic about it from a cryptographical security point of view. It's pretty straightforward and we'll review it briefly. What is nice about that protocol is that it is relatively elegant and it relies on off the shelf cryptographic primitives to achieve something which may actually work in practice. The proof that it may actually work in practice is the fact that it was used as the basis for a number of implementations, in particular I am aware of three implementations in the recent couple of years.

One was at MIT by Herschberg which I'll refer to in a second, another was at Princeton and the third one was at Washington University, St. Louis.

Joan Feigenbaum: It was implemented for elections?

Reply: Each one of them was implemented for elections within a campus-wide context of students.

Joan Feigenbaum: So it is for elections, it's not an embedded sub-protocol in something like a distributed system.

Reply: No, it is meant particularly for elections.

I want to review briefly the original FOO protocol and what it does. As I said it's a nice elegant simple protocol as far as some of these protocols go but it achieved its purposes. I will talk about a couple of weaknesses after I discuss the protocol but in general it's an elegant design of protocol. So the environment we are assuming is of N voters where N can be quite large, it's a system which is supposed to work in a widely distributed scaleable environment. So we have two voters and we have two servers which actually handle the elections. A is some sort of administration server and C is a counter server. Each voter is involved in an interaction with each one of these servers and each voter once he or she comes to vote performs a couple of preliminary operations, namely fill the ballot and enter it on her PC, perform a bit commitment (using something like Naor's scheme) and then on that bit commitment forms a blinding of that ballot, again using conventional blinding techniques which most of you will be familiar with. The result of these operations is sent to the administrator which has a pre-defined list of eligible voters and what the administrator does is sign the message that it received from the voter after it checks the eligibility of that voter to vote and the fact that voter has not voted before — the usual checks. Because of the conventional properties of the blinding features the administrator signs without knowing what the content of the votes are, sends the blinded signature back to the voter which she unblinds, using the blinding factor which she used originally, so gains as a result the vote signed by the administrator and she is now in a position to actually perform her vote with the counter C at the second phase of the election.

The election consists altogether of three phases. So at the second phase the same voter from the same station now communicates with the counter to deliver the vote via an anonymous channel. So there is an anonymous channel which allows the voter to maintain her anonymity with respect to the message transfer characteristics. Since the voter committed, the signing of the server was on the commitment of the vote, so on the second phase the voter sends the commitment to the counter. The counter records and publishes all the commitments it gets at the second phase and once the second phase is completed, at that point the counter has published all the commitments of all the voters, each voter is then at the third phase in a position to open his commitment to send the real vote to the counter again by the same anonymous channel and at that point the counter, after making appropriate checks for each vote, mainly the vote versus the commitment, publishes the list of votes with commitments. So the nice properties of this election scheme are that it uses pretty standard cryptographic primitives and that each voter can check that his vote was recorded correctly, all the voters can check that no entity which is not on an eligibility list has been allowed to vote and each one can check the tallying of the vote is consistent with the individual votes.

There are a lot of possibilities for each voter to make sure the vote is performed correctly. Drawbacks of this approach, well I list some in the position paper. If one looks at the details of the protocol then some of these messages are underspecified. For example, the message in the first phase between the voter and the administrator is underspecified and may result in the situation that the administrator can take another signed message from a different context altogether and claim that this was a vote. I remind you [a vote] is a blinded string of bits and so the administrator can play games of that nature, but this can easily be fixed and that is not what I am really concerned about here.

Another type of potential problem with that protocol is the fact if things go wrong, in particular for example, if the administrator signs something else, the administrator may have an interest to prevent the voter from completing her vote correctly, if the administrator signs something else or deviates from the protocol, according to the specification of that paper the voter needs to reveal the blinding factor and prove that something went wrong here which may lead to all sorts of results which are inconsistent with the goals. In particular, if the VI reveals its blinding factor obviously at that point that original vote cannot be used by the voter because everybody would be able, using that blinding factor to, associate the vote with the voter. Consequently in order to complete the process the voter would need to start another session with the administrator allowing the possibility for the voter to lie about the response from the administrator and thus try and gain another extra vote inconsistently. Again that can be fixed relatively easily, it's more of an oversight than a real and significant bug in that protocol.

What is significant, however, in my view, is the fact that the voter has to stay on-line, or to have interaction with multiple servers for three phases of the protocol. In particular, the voter may be in a position to complete the first phase and for some reason be prevented or not be able to complete the second and third phases, which are likely to happen the next day, depending on the length of the interval of the elections the voter has to have interaction with these servers for a significant period of time. So that's obviously inconvenient but there is a significant security risk here, because the protocol is such that if the voter has not completed all the three phases of the protocol the administrator is in an excellent position to forge, to use, each voters rights to cast bogus votes and thus cheat the system, so the higher probability the voter doesn't use the votes, the higher the opportunity for the administrator to cheat and cast bogus votes on behalf of the voter.

The system employed at MIT by Herschberg as part of a master's student project and which is actually being planned to be deployed in a few weeks as a real system for running the elections of this student body, is again a very elegant re-implementation of a system based on similar principles but includes three improvements on the original full protocol. In particular, one takes a step further in improving that concern which we discussed, namely, the fact that the voter has to stay to interact for a long period of time with multiple servers. So the Herschberg full protocol essentially consists of similar structures, namely voter administrator interaction, blinded commitment of the ballot returned signed to the voter, but then at that point the voter performs one more step and sends the committed vote to an anonymous server and the anonymous server keeps all these votes until the end of the election, and the anonymous server itself transfers the vote to the counter server at the end of the election. So the nice feature of this type of an approach is that the voter does not have to do the checks herself, but rather completes the first phase, checks the result of the first phase, sends the vote to the anonymous server and at that point can go home assuming that her vote was cast correctly and that she has completed her democratic obligation to participate in the process. There are a couple of potential weaknesses in that protocol. One of these is that because of pragmatic reasons the interaction between the voter and the administrator is done not with digital signatures but with a shared password between the voter and the administrator. Which actually has a very significant implication for the ability of the administrator, not only to vote and to change votes on behalf of the voter, but much more significantly to vote on behalf of voters who didn't bother to show up and claim that they did show up because the whole proof of the voter's vote is based on a shared password. Another problem is the fact that the voter once he has submitted the vote to the anonymous server may go home, however, it's not clear according to the protocol specification that the anonymous server is in a position to verify that the counter has not deleted votes that were sent to it. In particular the counter, once the votes are opened and sent to the counter, may be in a position to delete votes which aren't favourable to his opinion and his view as to who should win the election. So although it actually shortened the amount of time that the VI, the voter, has to stay on-line to interact with the server, it has opened the voter to significant exposure, since under some circumstances her vote may be dropped without that voter knowing about it.

So finally the version here is based again on similar principles but with slight reworking of the details, and in particular on the use of this notion of optimistic trust aspects of that protocol. In other words with the design of protocol based on the same structure which would complete correctly, I would claim, if everything goes to the specification, but in any kind of deviation a passive investigator denoted E here, would be in a position to receive a wake up message from any one of the participants who discovers the deviation and be in a position to complete the investigation of the execution of the protocol with enough evidence, not only to ensure that the correct properties of the protocol are maintained, but to point the finger with convincing evidence regarding who the cheating party was. So again in principle (details are in the paper) the structure of the protocol is a single interaction of the voter with the administrator and the administrator at that point publishes an acknowledgement, which essentially contains a piece of information which guarantees to the voter, not only that his committed vote was received correctly, but if some inconsistency happenes with that vote from that point onwards there will be enough evidence to point the finger either at the administrator or the counter by involving the investigator. Furthermore, if the investigator itself is corrupt there will be enough evidence in the system to show that the investigator was trying to cheat the system under the assumption in this case that at least, that at most one of these guys would be corrupt, not more than that. If we are concerned about the possibility of more than parties either A, C and the investigator, if we assume that any two of these may be corrupt, we will need to extend the protocol to include more investigators using schemes related to thresholds, approaches along similar lines.

OK so the administrator publishes the acknowledgements. There is only one voting phase to this protocol, and at the end of the session, the administrator sends all the votes to the counter, so the administrator also serves here as an anonymous server in effect, because there is no way to associate as far as message traffic between the first message here and a message to the counter, and the administrator is not in a position to open the votes, so the administrator is not able to tie between a vote and a voter, and at the end of the day the counter receives all the votes, opens them, publishes the result and because of the way the protocol is specified both the counter checks the consistencies of A's publication of the acknowledgements and A checks the consistencies of the votes that the counter published, thus each one makes sure that the consistency is ensured and if it is not ensured each one of them can invoke the investigator who would complete the protocol while guaranteeing the correctness of the results and without under any circumstances being in a position, or putting any of the other parties in a position, to tie between a voter and a vote.

OK to conclude I'll throw in this headline which appeared in the MIT newsletter two weeks ago, again with a different election scheme which was held on campus, the trust betrayal there was not one of the more exotic, getting to political power by factoring a large number, but slightly more conventional use of spamming, however, it struck me as an example of the kind of problems we're trying to address here.

Using this kind of optimistic view to life, optimistic trust, while [requiring us] to find the fallback options, allows in general and in particular example to improve the performance of the protocol, because in particular we have less messages, each voter is limited to a single interaction, and at the same time contrary to the common trade-off between trust and performance we are actually improving the security because the single interaction of the voter with the server means, in this example we demonstrated, that there is less opportunity for the server, and in particular the administrator, to exploit the fact that something was not completed and so we lessen the opportunity to really cheat as a result of the fact that the voter had a relatively big burden in completing correctly and ensuring correctly the results of the elections.

What still needs to be done, and is not included in the position paper, a more comprehensive full description of some of the aspects and some analysis which would actually make the claims that we're hand-waving here more solid. On-going work is focusing on applying similar principles of optimistic trust and the use of investigators to other types of electronic commerce protocols and this notion of investigators or passive entities, to optimistic trust, is in a sense, in some respects self enforcing because the parties, which we assume already optimistically that they are likely to perform honestly, knowing that there is ability not only to detect the deviation but to point the finger at them, are more likely given the notion of investigators in the protocol, to behave according to the specification. So there is a loop here which is self enforcing which incidentally brings up an interesting sort of business model challenge namely if we look at commerce protocols and we incorporate such notion of investigators we are assuming that they will never be involved or very seldom be involved, just by the fact that they exist there, so who is going to pay for them. I mean the point being that somehow their very existence is to be there in order not to be used, and so it causes a sort of a question mark in order. If one thinks about it from a commercial point of view, what is the incentive for somebody to provide such a service which is not likely to be used just by the fact of its being there.

So obviously there are solutions such as subscription and insurance policies, etc. and these are the topic of a different talk, but this kind of challenge is an interesting angle.

Michael Roe: It struck me that that kind of situation where you have that investigation after the fact occurs or ought to occur with almost all protocols that have any non-repudiation. You've got this notion that these digital signatures you check in-line in the protocol but you're saving away an audit log and if you were doing it properly there would exist this ability to get these things out of the audit log together with all the associated stuff across to a third party's system and have it verified there. Of course in reality we know that this stuff never gets built but, a similar thing when we're looking at SET, you know the Microsoft transaction protocol, there really ought to be this tool whereby if somebody does dispute their electronic payment you can get this stuff out of their system and do the independent check and there's a question as to who pays for it. The natural place for this to be seems to be the conformance testing tool. If you're in a world that you actually do a check, somebody's already got the implementation of a protocol, what they do to get their stamp "yes this is a correct protocol", we do this, we deliberately dispute a transaction.If it produces the right sequence of bits that satisfy the conformance test tools for these test disputed transactions it gets its tick for this is a genuine good implementation, and because you have to pay for conformance testing anyway you can smuggle the cost of building these arbitration tools in with the cost of testing it.

Bruce Christianson: The other great advantage of this holistic approach is that it allows you take the entity that does the prior on-line authorisation test out of the trust envelope completely. You don't care if that mechanism is no longer reliable so long as in fact it works most of the time.

Reply: Not convinced.

Bruce Christianson: This is an important point, it means that, for example considering transactions you can have your favourite locking mechanism and treat it as a soft locking mechanism so if it ever deadlocks or breaks the security policy you just break the locks and carry on. But you know that most of the time you can prevent the sort of conflicts you're trying to prevent. You don't have to trust the mechanism that's doing access control, you don't care, so long as you can drive the number of incidents that you have to clean up afterwards down to the point where you can bear quite a high cost per incident.

David Wheeler: I didn't quite follow in detail everything, I'm not quite sure what you mean by publication, but could you do the following, that when a voter votes by whatever mechanism you have he supplies a random number and then you have lists of votes for each candidate with the appropriate random number then a voter can verify as you say that his vote was actually counted?

Reply: That actually is related to the idea which I didn't go into the details of, however, you have to be careful, it's not just a random number. What I am proposing is a random number encrypted with the public key of the counter because you have to make sure that on the one hand the random number is published, on the other hand you want to make sure that the counter is able to associate the random number with one of the votes it receives and thirdly you want to make sure that nobody else, in particular not the administrator, is in a position to publish the association between the name and something which the counter would be able to trace back the vote to the voter. So you're perfectly right, in what I'm proposing is something which is based on the same principle but looks at some of these issues in particular.

Frank Stajano: This thing you mentioned about, we put these investigators there and because they are there they scare people off and so they will never be used and so on. This reminds me of the same argument for, you know, big weapons, ballistic missiles, and so on, it's the same sort of balance isn't it?

Reply: Right, right, but there you and me are paying. It's not clear that we want to pay for these electronic commerce protocols, but it's an interesting argument.

Frank Stajano: But how expensive are they compared to the rest? How expensive in the context of building this system?

Reply: Well that depends, we're talking about a real system which has to be scalable and the cost is maybe significant, however, whatever the cost is, even if insignificant, who's going to incur it. You need some sort of economic incentives for the player and again there are solutions and I mentioned one, you can think of things like purchasing in advance security insurance policies or subscription models which would solve that particular problem but it's interesting if you think about micro payments and paying as you use and these guys will be left broke very quickly because whatever the cost is they wouldn't be able to refer it back so it's a losing proposition.

Frank Stajano: As Roger said on the first day, you just have to balance it against the cost of having fraud and if your claim is true and these things are reducing fraud, then all the costs that you would otherwise pay in the fraud goes towards that.

Reply: Right but again, I'm thinking about the economic side not so much in election schemes which obviously is a particular example but in a more general economic commerce interaction whereby you have a buyer or seller or whatever, some sort of economic interaction, at that point each one of the parties is an interested party in particular it may gain advantage at the expense of the other party. At that point you don't want one of the parties to sponsor this, that has to be an objective entity by definition. Consequently you can't expect one of the parties to incur the cost and still require the other to trust them. I mean he may or may not trust him but it would not be reasonable to trust a party which is not impartial so the impartiality brings in the economic issue, you'd need to have somebody with a third party in a real sense and would have to somehow exist in a real business economic world. How do you do that?

Carl Ellison: I don't see too much problem there because there's a function that this investigator could perform that people need and this is a secure off-site audit trail logging. And the investigator could sell the service of providing an inaccessible audit trail.

Reply: That's certainly one interesting direction although again why do they need to investigate logs, one of the reasons is something being wrong and we are claiming in principle because of the self-enforcing property here, things are not likely to go wrong because parties will behave correctly which again would ...

Carl Ellison: But for any protocol involving banks you have to keep an audit trail.

Reply: OK, so you can piggy-back another related functionality and justify it economically. I agree, that is certainly an interesting possibility, sure.

Bruno Crispo: Can we split the investigator to each party?

Reply: The investigator, as I said, this is a particular example in which there is only one investigator, in reality ...

Bruno Crispo: No, no, would the investigator exist just if there is a dispute for example and all the parties cooperate and then there is this investigator?

Reply: Oh, if both parties cooperate to cheat?

Bruno Crispo: No if the investigator exists but is a virtual thing, distributed between all the participants.

Reply: Sure that could bring us closer to the privacy I mentioned in which there is no investigator but each one of the participants is involved in a very, very complex protocol. Sure you can do that but things become very quickly unrealistic when you talk about more than a very few number of voters. So the answer to the question, yes you can do that with some things, how realistic they are is a different question.

Wenbo Mao: These are all not a real solution. One of the ways to do this is called a public verifiable board, which means there is verifiability of the board, and the verifiers are all voters, every single voter.

Reply: Right, well one example is publishing it in a web page or a voting board and that's the reason the FOO protocol described is based on a similar principle, but we show that it's not enough, part of the problem of ensuring your vote is counted and that everything tallies is not enough because quite significant things can go wrong within that context. Plus you're putting another burden on the voter.

Virgil Gligor: I have a comment not a question, so it's outside the rules. What about this. What if you statistically verify the validity of the result, not necessarily that every single vote counted was counted right. You look at difference between yes and no, the global yes and no, and statistically verify it.

Reply: Let me answer to something that is closer to home. Again the Binjamin Netenyahu against Perez election of two years ago, I still remember Perez opening the champagne bottles after the election based on a TV poll and then the results being reversed by a very small number.

Virgil Gligor: But that's different, that's not verification, that's prediction.

Reply: No, but I'm giving you the example of a very close election which can turn on a few votes.

Insider Fraud
(Position Paper)

Dieter Gollmann

Microsoft Research Ltd
Cambridge, United Kingdom
diego@microsoft.com

Abstract. Frequently, inventors of an attack desperately try to find reasons why the victim of the attack should have initiated a protocol run with an *intruder* when it is blatantly obvious that there is no intruder anywhere to be seen but there is a misbehaving *insider*. Security models where the antagonist is an insider are much more relevant to the electronic commerce scenario which today drives much work on security protocols and cryptography. This is another example of a general problem in security. Too often, the concepts used to discuss security do not fit the security issues we are trying to address.

An adequate description of a problem is usually the first step towards its solution. Conversely, when we cannot describe a problem properly, what hope can there be that any solution we come up with will be adequate? To describe a security problem, we have to state

– the (threat) environment in which the problem should be solved,
– the goals that should be achieved.

Then, we can investigate whether the protocol meets its goals and whether the goals are adequate for the preceived threats. Unfortunately, research in security rarely follows this route. Of course, it is a valid exercise to examine all the properties a protocol may achieve but arguments of this kind should be the exception rather than the rule.

We use authentication to illustrate the deficiencies in many current discussions of security. In a previous paper [2] we did address the second point made above and showed that the term 'entity authentication' has a number of different interpretations, all of which are reasonable so that there is no 'true meaning' of entity authentication. In this note, we will draw attention to the way the threat environment influences our perception of the expected behaviour of the parties executing and attacking a security protocol.

1 Intruders

In most descriptions of 'attacks' against cryptographic protocols, the attacker is presented as an *intruder* who tries to interfere with the communications between some other parties so that their security is compromised. For example, [4,5] use the symbol I to denote the intruder whilst $I(A)$ (or I_A) stands for the intruder impersonating the party A. In [9], the attacker is a *penetrator* E and E_A indicates that the penetrator masquerades as party A. In [7] and [8], the attacker is a *spy*.

This attack model has a long tradition. It still has a firm grip on security researchers, who feel obliged to explain any attack within its framework. We will use Lowe's attack [3,4] on the Needham-Schroeder public key protocol [6] to make our point. This protocol is based on a public key encryption algorithm. By encrypting a challenge R_A with the receiver's public key P_B, the sender A can be sure that the intended receiver B has to reply to the challenge before it can become available to anyone else. In a simplified description of the protocol, mutual authentication proceeds as follows.

1. $A \rightarrow B$: $eP_B(R_A||A)$
2. $B \rightarrow A$: $eP_A(R_A||R_B)$
3. $A \rightarrow B$: $eP_B(R_B)$

Here, B determines the public key P_A from the address field in the first message. Some security properties of this protocol have been proven in the BAN logic [1], e.g. that A and B believe R_A and R_B to be shared secrets. In [4], the following 'triangle attack' was presented.

1. $A \rightarrow I$: $eP_I(R_A||A)$
2. $I(A) \rightarrow B$: $eP_B(R_A||A)$
3. $B \rightarrow A$: $eP_A(R_A||R_B)$
4. $A \rightarrow I$: $eP_I(R_B)$
5. $I(A) \rightarrow B$: $eP_B(R_B)$

We can examine in which circumstances such a sequence of events constitutes an attack (see [2]) and why such an attack can co-exist with BAN proofs that derive properties violated by the attack (see Section 4). At this point, we are more interested in a further issue raised by this example. An explanation of this attack faces an awkward problem. Why should party A start talking to an intruder in the first place? Hence, you will sometimes hear that 'A is tricked into initiating a protocol run with the intruder' or that 'the intruder starts a false session with B'. Of course, this problem completely disappears when we treat I as a misbehaving insider.

Given that statistics on security breaches typically suggest that insiders are responsible for 70 - 90% of all incidents, it is only reasonable that we adjust our security models to deal with insider fraud. In this note, we will contrast the old and the new security paradigm and start to examine how insider fraud affects the validity of current approaches to protocol verification. In particular, we will mention the role of secrets in authentication protocols and argue that insider fraud and man-in-the-middle attacks should be treated as different types of attacks.

2 Security Paradigms

We have inherited our old paradigm from communications security. Communications security addresses the situation shown in Figure 1. Two entities A and B, which trust each other, communicate over an insecure channel. The antagonist is an *intruder* who has full control over this channel, being able to read, delete, and insert messages. The two parties A and B want protection from the intruder. This traditional view of who is friend and who is foe is reflected in the axioms of many verification tools for cryptographic protocols, which assume that A and B will behave according to the rules of the protocol and only consider the effects of the intruder's actions. The most notable example is the BAN logic [1], where such assumptions are made quite explicitly.

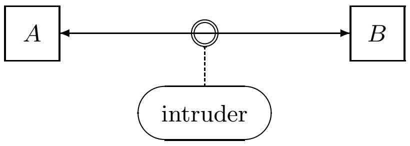

Fig. 1. Communications Security

A new paradigm is suggested by *electronic commerce*, where a customer enters into a business transaction with a merchant. Although both parties do not expect the other to cheat, disputes are possible. Customer and merchant thus have reasons to run a protocol that does not assume that the other party can be trusted in all circumstances. In our new paradigm, the antagonist is a misbehaving *insider*, rather than an intruder, and the third party in Figure 2 is no longer the intruder but may be a trusted third party, for example a notary or an arbitrator.

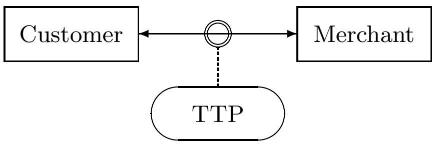

Fig. 2. Electronic Commerce Security

3 Insider Fraud - A Toy Example

To demonstrate that we do not have to invoke the existence of an intruder to have an attack, we embellish the Needham-Schroeder public key protocol so that it includes information retrieval and payment for the information received.

1. $A \to B$: $eP_B(R_A||\text{info request from}A)$
2. $B \to A$: $eP_A(R_A||R_B||\text{the info})$
3. $A \to B$: $eP_B(R_B||\text{info ack+payment})$

Lowe's attack can then easily be explained as insider fraud, without having to assume that A is tricked into talking to the attacker. Assume that A wants a particular piece of information and will pay the provider of the information. B is a *bad* party pretending to have this information, while C actually has the information but would not pass it on to B. The insider B is still able to defraud A and C. To do so, B re-encrypts A's request and passes it on to C, C replies to A, and A acknowledges the receipt of the information, together with some payment to B.

1. $A \to B$: $eP_B(R_A||\text{info request from}A)$
2. $B \to C$: $eP_C(R_A||\text{info request from}A)$
3. $C \to A$: $eP_A(R_A||R_B||\text{the info}))$
4. $A \to B$: $eP_B(R_B||\text{info ack+payment})$

By the time C starts to complain to A about the missing payment, B has cashed in and started trading under a different name.

4 Keeping Secrets

It is not at all surprising that the threat environment has an impact on the way we reason about protocols. This point becomes evident when we investigate the apparent contradiction of BAN proofs for the Needham-Schroeder public key protocol and Lowe's attack against it. The important observation about Lowe's attack is the fact that B is asked to handle a 'secret' R_A but is not really compromised if it discloses the secret to a third party. This is fundamentally different from protocols where the secret is a key that authenticates B. In the latter situation,

- B has good reasons to keep the key secret, and
- once B has disclosed the secret, the protocol can achieve nothing.

It is therefore a reasonable and necessary assumptions that secrets will always be kept. For example, the BAN logic assumes that secrets are not revealed during the protocol run. The BAN logic uses the symbols

- $A \overset{Y}{\rightleftharpoons} B$, to denote a secret Y shared between A and B,
- $\langle X \rangle_Y$, to denote a message X combined with a formula Y.

The BAN axiom for shared secrets is is

$$\frac{A \textit{ believes } A \overset{Y}{\rightleftharpoons} B, A \textit{ sees } \langle X \rangle_Y}{A \textit{ believes } B \textit{ said } X}.$$

If Y is a secret shared between A and B and if X is a message constructed using the secret Y, then A can conclude that B sent X. This axiom relies on Y being a secret known only to A and B during the entire protocol run. If Y were revealed, there would be no justification to attribute message X to principal B.

In [1], some properties of the Needham-Schroeder public key protocol are proven using this axiom and it is suggested that the secrets R_A and R_B could be used to form a session key for subsequent message protection. The idealisation step in the proof and the suitability of the session key derived in this way again rely on the assumption that secrets will always be kept.

When parties have little to lose from disclosing a secret, this should be reflected in our security model. Temporary shared secrets (one-time secrets), i.e. secrets that are only protected during the transmission of a message through cryptographic means but may be disclosed by the recipient can play a useful role in security protocols. They can be included to identify the recipient of a message but do not identify the sender of the response received. For example, a valid response to a challenge encrypted under the recipient's public key implies that the receiver did use its private key to obtain the challenge during the protocol run. The response may arrive in a message sent via some other entity or even in a broadcast. This feature of temporary shared secrets is captured by the axiom [2]

$$\frac{A\ \mathit{believes}\ A \overset{Y}{\rightleftharpoons} B, A\ \mathit{sees}\ \langle X\rangle_Y}{A\ \mathit{believes}\ B\ \mathit{said}\, Y}.$$

If Y is a secret initially shared between A and B and if A sees a message constructed using the secret Y, then A can conclude that B did retrieve the secret Y. B can reveal this secret to other parties without invalidating this conclusion.

The BAN proofs made the 'mistake' of using an axiom for permanent – and pre-arranged – secrets to reason about temporary shared secrets established during a protocol run. At the root, there may well be again a problem of language. The meaning of 'secret' differs depending on the circumstances.

5 Insider Fraud or Man-in-the-Middle Attack

A precise language for discussing security should not overload its terms with varying meanings. This also applies to a classification of attacks. In a *man-in-the-middle attack*, the intruder intercepts the messages exchanged between A and B and inserts modified messages instead. The parties A and B are not aware of the intruder. For example, A and B could try to set up a secure session by exchanging their public keys at the start and using these keys to encrypt all further messages. The intruder intercepts these keys and replaces them by its own public key. All subsequent messages from A to B (and vice versa) will then be encrypted under the intruder's public key. The intruder can read and potentially modify the messages before re-encrypting them under the intended recipient's key. A and B are completely oblivious of the intruder's existence.

With insider fraud, the adversary is called into action by one of the parties. This party is well aware of the adversary's existence. Thus, we are not facing a typical man-in-the-middle attack and, for the sake of clarity, should use a different name.

6 Conclusion

When analysing security protocols, we do so within the context of the applications the protocols are intended for. When we find new applications for our protocols, the models that underpin their security analysis have to match the new application. Interesting challenges for protocol verification increasingly come from 'electronic commerce' while many security models are still rooted in a traditional communications security paradigm. We have tried to give some evidence for this claim, both in terms of the way attacks are described and in terms of the verification rules that are used.

There is a further general point this note wants to make. Conventional wisdom in system design advocates an approach where one proceeds from an understanding of the threat environment to the capture of protocol goals, and then to the actual design of the protocol. At each step, it is possible to check whether the new 'artefact' is consistent with the assumptions made at the previous stage.

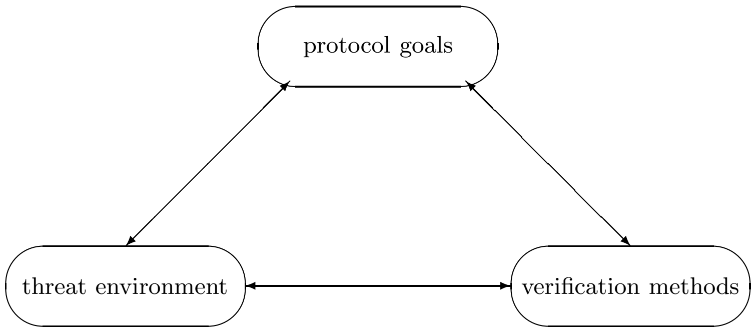

Fig. 3. Aspects in the design of secure systems

Quite alarmingly, in many cases security discussions proceed in exactly the opposite direction. They start with a security protocol, say an 'authentication protocol'. The goals and the designers' intentions are then reverse-engineered by inspecting the protocol itself (see e.g. [8]). Attacks are constructed to violate the goals thus derived. The threat environment is never really considered, or the standard communications security scenario is assumed by default, which amounts to the same.

Similarly ill-structured discussions currently seem to be under way on issues like certificates, delegation, or intellectual property right protection. As long as we start our investigations from technical constructs, like cryptographic protocols, and then try to figure out their 'correct' meaning, we will be disappointed and remain confused, because these constructs have different meanings depending on the application. We have to follow our own advice, consider security properties in the context of the particular application and describe the security properties that ought to be achieved in a language appropriate for the technical level at which the protection mechanisms are specified.

References

1. Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *DEC Systems Research Center*, Report 39, revised February 22 1990.
2. Dieter Gollmann. What do we mean by entity authentication? In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 46–54, 1996.
3. Gavin Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–133, 1995.
4. Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of TACAS*, pages 147–166. Springer LNCS 1055, 1996.
5. Gavin Lowe. Some new attacks upon security protocols. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 162–169, 1996.
6. Roger M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *CACM*, 21:993–999, 1978.
7. Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. Technical Report 443, Computer Laboratory, University of Cambridge, February 1998.
8. A.W. Roscoe. Intensional specifications of security protocols. In *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 28–38, 1996.
9. Paul Syverson. Adding time to a logic of authentication. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 97–101, 1993.

Insider Fraud
(Transcript of Discussion)

Dieter Gollman

Microsoft Research

Virgil Gligor: Presumably this has something to do with cryptographic protocols and money.

Reply: Not necessarily. It's a rather different background to the story. Back in February I was forced to propose something I would speak about at this workshop and really the only thing I dare to talk about would be some observations I had made about how people present analyses of cryptographic protocols, how people speak about cryptographic protocols. It seemed to me that we should be turning the attention towards the fact we also want to protect ourselves against insiders within the protocol. From looking at what I have heard in talks, it seems the point is worth repeating. It's a nice attractive title which has brought some people back to the room. Starting point: the public key version of the Needham-Schroeder protocol which has led to Gavin Lowe's attack. There exist BAN proofs on this protocol proving various properties roughly along the lines I have described here; the two parties believe that the nonces that are exchanged in the protocol actually constitute shared secrets. Martín Abadi has warned me don't call it a proof of correctness, there are proofs proving those properties and there is nothing wrong with the proofs.

So that was the starting point and along came Gavin and he presented an attack which I like to phrase in this form: A initiates a protocol run with a party called I. This party I now behaves strangely, re-encrypts the same message and passes it on to B. B treats it as a message coming from A and sends it back to A. A sees nothing wrong, everything according to the protocol rules, replies to I and that's how it finished. You can discuss whether it's an attack or not but certainly with respect to the one property I mentioned before that these nonces are shared between A and B, this is clearly not the case, they are now shared between all three parties. So with respect to this proven property this interleaved protocol run creates a violation. The problem people have with describing this attack is to find a good reason why A should talk to the intruder in the first place. I've heard it before, I've heard it from Gavin, I've heard it from other people and they say "Well, A is tricked into talking to the intruder". My point is there is no intruder. Why are we desperately looking for an intruder? I is an insider who behaves exactly according to protocol rules. You don't have to invent reasons why A wants to talk to an intruder or has to be tricked into talking to an intruder. Insiders can behave in strange ways. Perfectly normal, we know it. Typical statistics from security consultants on what are the sources of security incidents, Stew had one yesterday and the figure was 68, the breakdown quoted in my paper was 70-90, so if we put on our hats as security experts we know insiders are the main problem. We switch back to protocols and we're desperately looking for intruders because we cannot talk about security violations without seeing an intruder. Again in the paper I have briefly surveyed notations in various formal tools to analyse security protocols and predominantly they want to talk about an intruder because that it is the “right” way to talk about cryptographic protocols.

One of the reasons for this is a shift in paradigms which we are not prepared to acknowledge intellectually. We start off from this old communications security paradigm. A and B trust each other. The intruder sits on the communication link and can do all sorts of nasty things, intercepting messages, changing messages, deleting messages, and inserting messages. What these protocols for communications security do is protect A and B from the intruder. Perfectly reasonable goal for a communications security program. Now you move more and more to electronic commerce. You find many papers now on security protocols who feel obliged to mention electronic commerce to make the point that it should receive more funding and should receive more attention. In electronic commerce, I have now these more colourful As and Bs. You have merchant and customer. They are definitely not partners who necessarily trust each other completely in the same way as complete trust exists in these communications. It would be wrong to say they are mistrusting parties, but as in Rafi’s talk before, you know sometimes things go wrong, maybe because someone really tries to beat the system, maybe simply because something accidentally happens in communications or in the implementation. Things go wrong, you have to recover from this state, you need some process of dispute resolution and in that scenario, the protection that B wants is from misbehaving insiders and if you want a third party monitoring this traffic it may be a trusted third party, an arbitrator, investigator in the last talk. So paradigms shift, the role of the parties shifts, the underlying environmental model shifts, but I still would claim most of us, all of us here, if we asked to explain security protocols to someone who said “This is something new to me please explain”, we’d latch on to the communications security paradigm and explain what we’re doing in these terms. Language and fundamental patterns of thinking lead us astray. To make the point again about what I would prefer to call an insider attack, I’ve embellished the protocol. I’ve said the first message from A to B is a request for information to some information broker. The information broker provides the information and somehow confirms it’s a reply to this request, it’s from me, because the information broker gets some reward in the third message for the goods delivered. Now B is the bad insider, setting himself up as a content provider, but he hasn’t got anything to offer. C has the information. C knows B is a competitor. C wouldn’t on its own give away information to B so B can make money with it. A goes to B because B has advertised his services and asks for a certain bit of information. B plays the game as in Gavin Lowe’s attack: passes it on to C, to C it now appears as a completely innocent request from A, replies to A, A receives the answer to the query it asked B for, so the electronic brownie points go back to B. Insider attack, no intruder. That’s what I want to call it. Another footnote here: I’ve seen this type of an attack being categorised as the man in the middle and I don't like it. You are again using the same term for very different types of events. There are man in the middle attacks where A and C are the victims, and they don't know that anyone is sitting in the middle intercepting the traffic, playing around with the traffic. Here A wants to talk to B. B is open, it's not someone hiding on the network and playing silly games, So my plea is: call these insider attacks too. Sounds more attractive.

Michael Roe: Because A believes they're talking to B they don't believe they're talking to C. If A believed they were talking to C it might be more like the traditional man in the middle attack.

Reply: Exactly, if A wanted to talk to C and B somehow inserts itself and gets the reward intended for C, then yes I would call it a man in the middle attack. If A wants to talk to B in this case and gives the reward to B then A has been fooled, C has been fooled, B makes a run with the brownie points.

Bruce Christianson: A man at the far end attack?

Reply: So, to conclude this part of my talk, what I have seen in formal verification protocols, security protocols, cryptographic protocols, you have in the bottom right hand corner verification methods, various proofs developing the tools and showing what they can do in the analysis of security protocols. I see something alarming, something extremely strange, I see it again at this workshop. I see people look at the protocol, look so to say at the implementation and reverse engineer the specification, so what could the designer have meant when using this cryptographic algorithm? I think this is an extremely strange way of thinking. In terms of software engineering, that's contrary to everything we preach, nice top-down approach, you have the specification, you have the implementation, you show that the implementation meets the specification. Here repeatedly you see people starting with the protocol and then ask what could this protocol have been meant for. The most blatant piece of evidence is a paper Bill Roscoe gave at the foundations workshop a couple of years ago on intentional specifications of security protocols, where the proposed strategy was to start from the protocol, derive the specification and then show that the protocol doesn't meet the specification.

Carl Ellison: Does this surprise you? This is the way software's documented nowadays.

Reply: It doesn't surprise me.

Bruce Christianson: It's not just the way it's documented, it's the way it's debugged.

Joan Feigenbaum: What's bad is when they can't answer, when you say to the implementers, so why are you encrypting this, what are you trying to protect, and they have no answer. If they have an answer then it's not so alarming. It's when they don't know that you have a problem.

Michael Roe: In the normal non-security case you get called in to consult because some guy has written this piece of code and left the company and nobody knows how it works and there are you reverse engineering and trying to write a spec for it. And as security consultants we find ourselves doing the same thing in security applications, there is this piece of crypto protocol, nobody has a clue what it's for or what it's doing and you are called in as this highly paid consultant to try and work out what the spec should have been in the first place. And because we do this for customers we tend to get in the habit of doing it, but it shouldn't be encouraged.

Larry Paulson: All these Internet Drafts I've been looking at are the same.

Reply: Yes I agree, they're awful.

Bruce Christianson: The other alternative is to say well I'll do the spec beforehand when I'm not sure how I'm going to implement it and then I'll leave the real problems of actually getting the security policy to the implementer who will make arbitrary decisions that have far reaching consequences that I haven't thought about.

Reply: I think you're coming into this interesting discussion and unfortunately Peter isn't here, SC 21 and SC 27 keep beating themselves up, one groups saying lets start with nice service definitions and then define the mechanisms for the services we have defined, the mechanisms people shouting out saying you're defining services that can't be implemented, what are you doing? And vice-versa.

Bruce Christianson: If a spec helps you get the implementation that's very good but I want to see that you have an implementation that works before I'm willing to look at your spec.

Bill Harbison: Using the term "works" loosely there I take it.

Bruce Christianson: It does something and it might be interesting to break.

Reply: I observed this is going on and it isn't helpful and again I totally agree, I've read the RFC on Kerberos and I wouldn't recognise Kerberos from the RFC. I have seen it in industrial protocols which get lost in implementation details, key length, which choice of algorithm, how exact did you create this derived key and that derived key and I see strange things in the protocol and I develop theories, why these might be intended features and not mistakes made by someone who hadn't a clue what they were up to, but it would be nice and I think Larry said something similar, it would be nice if in a protocol you give a nice concise high level specification where you see what it's intended for and then going down to the more detailed specification you need if you want to have interoperable implementation of this. Why are you shaking your head?

Carl Ellison: You're just asking programmers to be able to think and to present ideas and mean this is asking a great deal.

Reply: Well you mention the word think. Back two years ago at the Open Conference they had this panel on security education. The panellists were people from industry being asked what would they expect universities to supply. The question was put to Roger Shell, "What do you want?" and he said "People who can think."

John Warne: I once saw a programmer do exactly what you're suggesting ought to be done. His name was Tony Hoare and he used to always approach the problem by a specification and then a refinement of the specification to an implementation.

Virgil Gligor: In defence of programmers who cannot think, one of the things we have to realise is that security was, is and will always be a secondary concern. First we have to deliver function and if we don't deliver function security is irrelevant. So what we often find is that in the specs that we try to see security we really find function with interspersed security ideas, or integrated security ideas, or security is bolted on. So sometimes it is very hard for a security researcher. Sometimes somebody who's mind is only on security doesn't understand function. It's very difficult to understand what this guy does. Again, I'm taking the extreme view here, I admit that there are lots of mistakes that people make and lots of free variables in security specs but at some point we have to admit that we are dealing with a secondary concern.

Reply: To be precise, I'm not so much accusing programmers of implementing things without proper specifications, I'm more accusing us as the research community of doing the same, and in some cases not realising what we're doing. To look at a protocol and think there must be an inherent, intrinsic, meaning to the protocol. If you look at Bill Roska's paper, it starts off in abstract saying "It has often been shown that protocols meet the wrong specification" and this is a meaningless sentence if you only look at the specification and the protocol, it becomes a meaningful sentence if you look at the application and say that this type of application and protocol doesn't meet the requirements, but to say there is a protocol and it meets the wrong specification, I repeat is in the true sense of the word meaningless and we're doing it within this community of security researchers so that is an observation.

So to come back to this diagram I think there are three areas we are looking at, verification methods, protocols and model of the environment and threats and quite a lot of work has been going on in the bottom right hand side, people are now playing around with defining goals, there is another panel at the Computer Security Foundations Workshop organised by Paul Syverson on what do we mean by authentication, and I think we're still insufficiently presenting the underlying and environmental threat model we are assuming. That comes back to what I said before. In our mind the risk is very strong this old communications security paradigm and even though we move in to applications in electronic commerce where the threats are different they still bug us. So that is the end of the regular talk and I have two excursions.

Excursion number one I can justify under insider fraud. We were talking about names and certificates and whether names and permissions are the same or something different. What struck me personally is there is very little idea about UNIX. I put it up to be corrected so I increase my knowledge. When I explain UNIX security to students I start off with the `/etc/passwd` file and what I'll find in there and then I'll say look there is this format, there is a user name, there is a UID, there is the real name of the user, so there is obviously an extremely close link between these UID and the user name, and if I look at how UNIX uses this particular UID it would do it for access control because for certain operations it would check where the request comes from because the UID would be used to indicate file ownership and then you can use it in access control decision. And you can use the same UID in your audit logs to say someone was logged in under this UID seems to have done something and the person who has got this UID is Dieter so we'll have a word with him about what's going on. Then you go on and read about how to implement your web server program on the UNIX box and the good advice they give you is to generate a new user, called the user `webserver`, with a new UID, a UID you don't use for anything else and then you give your normal users access to the web services through *setuid* to `webserver` programs. In the book I'm quoting it says we create a genuine user web server and I've put in the question mark because I think you suddenly using the same technically concept UID for something very different, you use it actually for a new form of access control, not really the access control you had in mind here where I was doing discretionary access control to my files, here you're doing something which I like to call controlling invocation. Users are allowed access to this web program only through this *setuid* to web server program. I don't think it makes a lot of sense to use this UID for auditing purposes, all you know is a browser has looked at a web page. Thank you very much. What you fail to see is that some versions of UNIX, ICL and also HPUX, then introduce another UID, a log-in UID in SCO, the effective UID in HP language which they use to keep track of who started all of this, and really if you read why they're doing it for increased assurance — auditing. So you have one and the same concept call it UID, of the same technical artefact within the UNIX system and you use it for very different purposes and if you're not careful you keep forgetting and that's something I like looking at, artefacts, which move around and have a meaning here, have a different meaning there. It's dangerous, it's convenient and I think the same is happening at the certificate level. In fact at some stage you have names which are really names. You have used the same concept, but you want to do something else and because it's convenient and because you've called it a name, you still call it a name, and you're not honest about what you're really doing.

So, as I said I am happy to be corrected and the last excursion, for which there is no excuse at all, is to put in an attack on insider trust which was triggered by the discussions on delegation we had. It was also triggered by the position paper Paul Syverson proposed for the Computer Security Foundation Panel where he is talking about authentication we were talking about delegation. I see a discussion in three different areas. Inside the system wherever that system is; outside the system; and at the interface. And with authentication we have authentication protocols and we argue about the relationship between keys and things like that. Then you have arguments why is an outside person linked to a cryptographic key and talk about the metrics and all this fancy stuff, smartcards, tamper resist models, and then this discussion on the interface, then you talk about which type of policies do we have on the outside. What I see happening is that the discussion moves straight through all of this, jumps back and forth, and again it's not helpful and that's it, as a convenient handover to the panel on summarising the workshop. Thank you very much.

Michael Roe: Would you feel happier about the UNIX case if you said that UID was a role? I mean it's the combination of the person and the role in which they're acting and the system administrator is acting as provider of the web service. This is pushing it as far as it can go, but ...

Reply: Yes, I would be reasonably happy to call this `webserver` user a role but the point is still you're using the same UID concept to do very different things. You use it to identify me as a real user who's gone to the system manager of the University, filled in a form, been signed by my Head of Department, this is a genuine user, allow him on the system. There is really a strong connection between that UID and me and I would expect if something goes wrong from some account, from that particular account, people would come to me and say have you done it, have you protected your password, have you told it to someone else. Again as Rafi was saying at Citibank, first thing you go to insiders before you check whether someone has broken in from the outside. Then you have this web server UID which does not have this connotation at all. You will never go to this dummy user `webserver` and ask him what have you done with your non-existent password. Completely different use of the same artefact.

Michael Roe: But if he does something wrong you do know who to go to enquire why it did that.

Bruce Christianson: But it's not a surrogate for him in the same way in which my user ID is a surrogate for me. It's different.

Michael Roe: It's "If all you've got is a hammer everything looks like a nail". UNIX gives you UIDs so you use them for everything.

Reply: Yes, that's what you've got.

Carl Ellison: And you don't enrich it with real user and current role, or a set of current roles.

Panel Session - Future Directions

Stewart Lee, Joan Feigenbaum, Virgil Gligor, and Bruce Christianson
Chair: Roger Needham

Stewart Lee: OK, I've learned a lot here. I have learned that delegation means many things to many people, like all of these words we use, and I would hope that somebody will launch a piece of research that will try and encompass all of the meanings, or a closed set of the meanings that delegation has. We have been using delegation as a sort of portmanteau, we could well have used some of the meanings of reliance for instance where we said delegation, and so on. And very often we were talking about delegation without worrying about the side effects: who is liable once we delegate, what happens to the liability when we do the delegation.

And very often we were delegating to what amounts to automata, very simple things, something like a printer server, and I'm not at all sure we'll ever get much beyond that. I delegate things to people but then I have a mutual trust relationship with the person I'm delegating. I don't have that kind of mutual trust relationship with things inside the computer so I pick only very simple things to delegate to. I think if we want to involve ourselves with delegation, then we might worry about protocols that are intended to do delegation, rather than delegation bolted onto protocols that were really intended to do something else. I don't know if there are any protocols that were inspired originally to do delegation and if there are, what should they do.

Finally I come back to something I said previously, that we have a system with a lot of stuff inside it and, because we all do research (in some context), we tend to concentrate on all this inside stuff because we get publications out of it and it's a difficult problem and so on and so on, versus producing something that is sufficiently simple that the typical brain-damaged user can use it versus who pays for it and what are the tariffs that we will use for it and how do we deploy those tariffs. These are all deep interconnected arrangements which we have to begin to look at.

Roger Needham: It's interesting for me to recall that when Abadi, and Lampson and Wobber and such people were putting together all this stuff on the subject of delegation they were in fact always talking about things like print servers. They were talking about rather down to earth questions like if I have got permission to access somebody's file (because it's Tuesday) how do I print it, what causes the print server to be able to have access to it. They were not thinking whatsoever about delegation in any human context at all, and I think they've been interpreted as saying something much beyond what they did say, in just the same way as Burrows Abadi and I were interpreted as having said something about belief when all we actually did was to give a name to a symbol so that we could pronounce it - laughter - when what the symbol was used for was defined by a set of rules. I have been accused of engaging in doxastic logic - laughter - I'm using very posh words, they sent me for the dictionary and I don't get sent for the dictionary very often and it actually turned out to mean a logic concerned with opinions. And we might think that the CERT crew on delegation and so forth have had their words used in an inappropriate context in the same way.

Joan Feigenbaum: We've heard a lot of interesting stuff about credentials and policies and trust and delegation and all these words that we haven't nailed down completely which actually is fine with me, I don't think we need to have some mathematically precise definitions for all these things, but I think it's really time to get started. This is unfortunately not the first workshop I have been at over the last couple of years where I've heard a lot of really stimulating and impressive talk about certificate mechanisms and trust and policies, and I'm pretty impatient, I'd like this stuff to get used. Rafi asked me, what would constitute evidence that what I've been working on with trust management is a good idea. I said, well I'd like to see this extrinsic, syntactic, application-independent notion of proof-of-compliance used in multiple applications, that conclude that it actually adds value or that it doesn't. That somehow really try to test this notion of an application-independent compliance technique. So I want to generalise that and say, for all the interesting trust mechanisms and trust policies and all the interesting stuff we've heard talked about during this workshop, I'd really like to see some serious commercial use of this stuff. So that in future we won't be having such an *a priori* and abstract discussion about what is delegation and what is trust and what is authorisation and what is it like. I think some of this will become quite clear when the stuff is used and it either works or it doesn't work or there's some grey area in between. If some of this stuff is used and it obviously does not work then we probably will have answered some of our questions. It's really time to use some of this stuff in some industrial-strength applications and I hope that happens before the next Cambridge workshop.

William Harbison: And then we can reverse engineer the implementation to get the specification and that will give us our definitions -laughter-.

Joan Feigenbaum: Yep, yep, I think that's often what happens. Some of the things we're talking about are inherently wrong or inherently misguided, I think that will become clear if there's an industrial-strength trial of this stuff.

I think Dieter's boundary between inside and outside is a really useful way to look at trust and delegation, I think that it dovetails very nicely with what Bill said much earlier, you have to have a trust policy if you want to authorise something that you don't understand. It only comes to the whole question of certificated policies and credentials when you need to trust something that you really don't understand and you don't have any direct authority over it but it needs to get done. There's a point when you don't understand what you authorise but you have to do it.

William Harbison: It was when I was talking about delegating trust and effectively saying we're passing off our own uncertainties to somebody else.

Joan Feigenbaum: Yes and that really has to be done, and it's done all the time in the world of commerce. But maybe in particular when you cross that insider/outsider boundary. Generalising this idea, of how can you authenticate someone from another administrative domain, how can you authorise something coming in from another administrative domain. Well that's when trust comes into it, that's when you really have to say, you know I don't understand this but I trust this third party or this set of third parties who do understand it.

I have one other point. Can somebody come up with some really interesting, important policy elements that should be tested by all of these methods. I've heard about unconditional delegation, limited depth delegation, a sort of threshold style delegation, at least two out of three, things like numerical limits on the amount of damage that could be done as a result of one of these delegations. What else is there, what else should your trust management system handle if it's really going to work in some commercial application. People say they would like to have tools that would help them analyse their policies. I would like to hear, in a not so analytical, not so precise way: what are your policies, what is it you need this stuff to handle, what are important security policies.

Roger Needham: I'd like to come back to the pre-conditions for the kind of industrial experiment Joan was talking about after the four panellists have each had a brief go.

Virgil Gligor: I have three points and a remark. The first one is about the new models of attackers that we have. In the past we focused on one type of attacker namely the intruder, the man in the middle, and the damage that the man in the middle could do, and we didn't focus as much on the insider. Although we did think of insiders, for example we considered party A in the Needham-Schroeder protocol, an insider, could launch a known-plaintext attack against party B. So party A had some advantage over an outsider and could launch such attacks. So it's not the case that we never really thought about insiders in the past. But there seems to be a fundamental distinction between the new models of attacks when we introduce also insiders, and the old types of attacks, the man in the middle attack, and that is that now we talk about multiple parties that may actually act in collusion. I have not noticed any discussion of collusion among insiders and possibly outsiders. For example you have three entities in electronic commerce: customer, merchant, bank, all of a sudden we have three groups that have different monetary interests and these interests can be independent or can be reinforcing each other and then we have collusion.

I believe that this kind of model of attacks will add an order of magnitude of complexity over the attacks that we considered in the past, so it's not only the notion of insider versus man in the middle or outsider, it's also a matter of collusion between various parties that have different interests. I think this is a great area of research with or without any industrial-strength applications. I would like to see some theory in this area develop. Now why am I pushing this. I have seen a class of problems in access control in the last twenty years or so where collusion caused an order of magnitude more analysis difficulty. Anything that has anything to do with leakage either for secrecy or integrity requires some sort of collusion. This will give inherently harder problems to deal with, at least theoretically, than problems that did not require collusion. In particular one problem that I worked on in the past was detecting collusion in audit trails and we showed that this is a fundamentally more difficult problem than auditing the actions of a single entity, which is hard enough in itself. So that's one area where I see some potential for profitable research with, again, no industrial-strength relevance at this point. Although I think that what Joan is saying is really fundamentally important.

Second area is this notion of trust. I view trust relations as dependency relations and in fact our design task in the past has been to minimise these dependency relations because usually dependency relations implied some sort of a liability: liability of doing the proof, liability of depending on some unknown entity doing something back to us. In general we try to minimise liability by minimising the degree of trust. Now it turns out that not all dependencies are bad, as a matter of fact we may have occasionally circular dependencies that from an analysis point of view are a disaster because you don't know where to stop or start testing, but from a practical point view they may not be. Professor Wheeler and I had a very brief conversation on the circular trust that you have between the merchant and the customer where in fact they both give the benefit of the doubt to each other because they both benefit from it. The point is that this circular dependency, circular trust, benefit of the doubt, may be conducted not from an analysis point of view, not from the point of view of an outsider trying to do testing or formal verification, but from a practical point of view, and so not all dependencies are bad even though most of them are.

OK, related to this point is the notion of benefit of the doubt trust, which is perhaps somewhat different from optimistic trust but occurs all the time in real life. Most trust relationships are not verifiable relationships, because if you can verify something you don't need to trust it, they are basically benefit of the doubt trust with some recovery against exceptional conditions such as cheating. So the notion of benefit of the doubt trust has to actually be investigated further from a theoretical point of view. I think there are notions of trust and dependencies and perhaps optimistic trust that all have to be put in a sound theoretical framework and I think that also offers some areas of research.

The last point is about delegation and then I have a brief remark. I believe that in practice delegation is a lot simpler than we make it sound, primarily because we deal with real systems where we actually delegate for specific functional reasons. That is not to say that it is an error to go and look at what delegation means in principle and examine all the parameters of delegation, because in the future it is likely that we will have application-level delegation that's required, not because of some system topology, but because of the fundamental need for delegation. So while we need to be realistic and focus first on system-level delegation, we ought to think in the future in these other areas of delegation that might appear at application level that are not necessarily realistic today. So delegation policy is, as was pointed out, a very big thing. When Lampson and I worked with delegation and we tried to define it, one thing that was missing from our logic was the policy component, or the logic attached to the policy so I think that is an area that ought to be explored in this sense of futuristic notions of delegation.

One last comment. We always assume that research into cryptography always simplifies the problem, in other words that cryptography eliminates some of those trust dependencies that we had. We have to be very careful with this because what I'm hearing more and more is that the use of cryptography introduces more dependencies than it solves. It may be these dependencies don't imply more trust but numerically sometimes you have more dependencies. So the question is, does cryptography really help or hinder this whole notion of trust? We have to have a really clear balance between the strength of the dependencies as opposed to the extent of the dependencies and really try to examine whether cryptography does provide a fundamental benefit. For example, we all depend on public-key cryptography of one or two types, basically RSA and Diffie-Hellman, universally we depend on this. What happens if this universal trust, or dependency, that we have on this gets to be violated, how do we recover from it? What if all of a sudden factorisation becomes easy and the whole world depends on it being hard. So I think that we ought to look at some of these fundamental dependencies that we have on cryptography and evaluate the use of cryptography further, rather than blindly going on with it and building elaborate things with even more dependencies on it than we have now.

Bruce Christianson: I would like to develop two chains of argument. The first is, by the end of this workshop I am now clear that I would like to see a better separation than we've made in the past between the design, or more usually the capture, of policy on the one hand, from the mechanisms that we use to represent and to enforce the policy on the other. I think we are always very tempted by this idea of developing some kind of Newspeak in which it's not possible to express things that you shouldn't want to do -laughter- I think we are going to have to reconcile ourselves to working in machine code for a little bit longer, until we understand what sort of things we should want to do and what sort of things we shouldn't want to do. I don't think we understand the issues well enough to be able to combine the two at the moment. So I support Joan in the idea that the mechanisms for representing and enforcing policies should be very neutral about what the policy is that they're representing, and they should be far more expressive than we think we need them to be or than we think is going to be good for us in the long run.

We also need to look at some specific application-based case studies, instances of what the commercial world would regard as very simple applications, and we need to try and understand what sort of policy we need for those and what kind of mechanism would be capable of supporting these very different requirements. At the moment we have a number of dots that we know we have to join and we're busy trying to construct an elephant in the middle with no evidence that there's actually anything in there at all.

So that's the first line of argument. The second is a matter to do with presentation to the outside world but it also affects our own internal goals. We've really got to re-examine our ideas about what the point of security is, what benefit security allegedly confers, because selling the benefits of security by saying that the effect of security is to make your system more secure is a clear loser. And it's certainly true that making systems, as systems are currently designed, more secure generally has the effect of making them more brittle, and it's usually perceived by the so-called legitimate users of the system (in other words, the insiders that we're trying to protect, from each other) as a systematic attempt to render the system less available or more difficult to use. And I think that we shouldn't see that as a symptom of the users not understanding how important security is, I think we should see it as a symptom of the fact that we haven't really thought enough about what it is we're trying to do. We have to understand security as a way of looking at system design which enables us to confer properties on systems that users think it *is* nice for them to have, such as robustness, integrity, availability, ease to recover from things going wrong that we don't expect to go wrong very often, the ability to do backups without having to think about it, and knowing that you've got exactly the information in the audit trail that you need for the bodies that you're accountable to, and even (looking at things like optimistic trust management) the ability to improve the performance of your system by moving stuff out of the trust envelope and by dealing systematically with different rare special cases.

I think that the move towards considering insider threats is the right move but it hasn't gone far enough. The insider who I'm most concerned about my system being able to protect me from is me. I want to be sure that I can't do anything that will mess up the system so horribly that I can't get back to where I was at teatime yesterday fairly quickly. We have an opportunity to recast a lot of what we do in that context and think about good ways of designing systems that we can then test by flipping the security on and off and seeing what falls over. So that's the second point.

Roger Needham: A slight thing I would like to add is that the goals a security policy supports may be completely unreasonable and nevertheless necessary. And I can give an example here in things like anti-piracy efforts. In relation to sale of, or sale of access to, content, whether it be content in the form of books, movies, pictures and that sort of thing, the owners of the content are extremely neurotic about getting paid for its use. And questions like possible development of electronic books would be determined by whether the anti-piracy measures match up with the unreasonable expectations of publishers who own the content, and this is irrelevant, one has to try to look for methods to support a stupid policy. That's alright, that's business life.

I would now like to ask for comments specifically on Joan's point about whether we need and whether we can actually get industrial-strength experimentation to study. Various things are in the way, people who do industrial-strength security have no confidence in their security so they won't tell you about it so you can't study it. Goldman-Sachs is a case in point.

Mark Lomas: I'd confirm that and say that part of our security policy includes a rule that says that people outside shall not see the security policy.

Bruce Christianson: Well, you've just blown that one -laughter-.

John Warne: I would like to refer back to my issue of closing the gap. Joan is suggesting now that industry begins to put these things into experimental practice but realistically you as the security community need to make observations of that practice and then to have an exchange of ideas to see whether in fact these things do work. This leads up to the notion of establishing a proper basis in which this exchange can take place and I don't know how that can be achieved.

Carl Ellison: Cybercash did publish its original protocol, made it available for formal analysis and it came out pretty well in that analysis and we're happy having done that. Maybe I'm being too cynical, but the thing that bothers me, in responding to Joan's request that I heartily support, is the fact that there are not already implementations of decent, strong, secure delegation of trust.

Joan Feigenbaum: You mean why aren't people using SPKI and SDSI and PolicyMaker and all this stuff?

Carl Ellison: Or the stuff that Butler was describing in 1991. The fact is, these are not brand new ideas. We are refining these ideas, we're adding to the field I hope, but this is not brand new. And yet the world has not beaten a path to our door and that fact is itself a data point, although not one that I like.

Joan Feigenbaum: What are the preconditions to get an industrial-strength experiment going? One would be a market demand for something that this technology needs. I hear a lot of people tell me that they do demand or at least desire various aspects of what I think good certification and delegation and compliance checking mechanisms could handle. Essentially they demand control over their own environment, they want their privacy, they want to say from whom they're going to buy whatever services they want to buy that require trust, they don't like it when you tell them that a bunch of decisions have been made for them already. But this demand that is voiced and this technology that might meet it somehow aren't meeting in the market place, there's been a market failure somehow, and I don't really know why.

Carl Ellison: Let me point out a couple of things. I don't know exactly the words you used Bruce but what I wrote down was that we have tended to deal with threat models that come out of our imaginations rather than threat models that we acquire from real users. And one of the sad things for me as a security person and cryptographer at Cybercash was the discovery that the company made with our initial protocol, that we had done just that. We had this wonderful protocol that we published and was analysed and blessed and we were happy with it and nobody really wanted it. So we have had to evolve our protocol to something people really do want which is nowhere near as iron-clad as what we originally produced. We have had to learn to respond to what our real customers want rather than what we thought they ought to want.

William Harbison: Yes, we've gone to customers and said what do you want us to do, and they said solve my problem for me, and we say what's your problem, and they say back to us, well you're the experts.

Peter Landrock: Yes, I concur with what Carl is saying. We cannot sell security, even at the research level. We are all trying to do it, but that's a fundamental mistake. You can sell secure applications, but you have to understand precisely what it is you want to secure. So a protocol in itself is not interesting, it's the context in which you deploy it.

Joan Feigenbaum: Absolutely, so that is what I'm asking. Can you tell me some good applications in which to test a research artefact that deals with delegation and trust and credentials.

Stewart Lee: Print servers.

Virgil Gligor: And that's all. The fundamental problem is that we factored out the security from real systems and applications, and we factored it out for academic reasons. We could keep it that way, we can live for the rest of our professional life dealing with very sophisticated policies, very sophisticated cryptosystems, systems that have nothing to do with reality. But if we really want to be relevant in any fundamental way we have to deal with systems, systems design, applications design, that provide function which has integrated security in it. So if we really want to do industrial-strength research and industrial-strength products we have to worry more about how to integrate security protocols in systems in a minimalistic way to solve some tangible problem that somebody can put their fingers on.

The big problem is that if we solve the most difficult security problems we cannot really demonstrate it all that well. Try to demonstrate security to an audience: I worked on two systems in the mid-80s, one was a B2 UNIX system called Trusted Xenix afterwards, the other one was a compartmented-mode workstation for the DIA or some intelligence agency. I was working within a group at IBM, so the IBM marketing folks decided to take these two systems and demonstrate them to real people. The compartmented-mode workstation had a lower grade of security but it had Windows and it had colour and could cut and paste between windows of different security levels and you could see multi-level security.

Joan Feigenbaum: You could see multi-colour windows.

Virgil Gligor: That's exactly what I should have said because the system was full of holes. The other one was really a decent system as far as security was concerned but it didn't have any of these fancy user-interfaces. Guess which people preferred? No guess, no contest, so part of our problem is that security on its own doesn't sell and we really have to sell it as part of our applications. Unless we start focussing our applications on systems we take the academic road.

Roger Needham: A few years ago I was on a committee at the Royal Society organising one of their exhibition evenings when they have everyone round in their dinner jackets and people demonstrating new advances and of course one of the great requirements for this is that what you demonstrate should be interesting to look at and to see working and so forth, and we had a great deal of difficulty in fending off proposals for a stand entitled "new methods of contraception." You can't demonstrate something not happening, and this is the trouble with security. Sorry.

Larry Paulson: A lot of this discussion reminds me in a rather abstract way of what happens in my own main field which is automated reasoning because a lot of grant money has been spent and no-one out there has ever paid any attention. And then Intel made this thing called the Pentium that couldn't divide properly and they lost something like half a billion dollars and I heard that they were then putting some money into formal verification and hiring quite a few people and I asked, are they putting in as much as they lost on the Pentium and the answer was, don't be ridiculous they can't afford that.

Roger Needham: Well they couldn't having lost it.

Larry Paulson: Well the point is that they are suddenly hiring people who they never would have looked at twice before and so actually what one needs in this field is more Citibank incidents which will make people sit up.

Joan Feigenbaum: I don't know whether you consider yourself part of the same general field as model checking. But I think that's a great research transfer, at least from the old Bell Labs' perspective and the IBM perspective. They have really somehow convinced people who do industrial-strength applications that model checking as done by the research community is something that they should care about.

Larry Paulson: Well I'm hoping that the soft drugs that there are now will lead them on to the hard drugs. They're proving addicts. John Harrison, for example, was recently hired by Intel. He doesn't do model checking but does the sort of proofs that takes months and are in absolutely horrible detail proving that a certain algorithm for the natural logarithm really is correct to 32 bits in the IEEE floating-point standard which suddenly people like Intel have decided is worth having done. And as I said you need more Citibank incidents. Maybe some of you could actually enrich yourselves at the same time - laughter-.

Roger Needham: We need sufficient incidents of security not having worked and we can't rely on the standards bodies for a steady input of wrong protocols which effectively we have done over a long period.

Carl Ellison: Final comment from me on Joan's request. If we, instead of being researchers, were a product development group inside a company, facing this request of yours, one of the suggestions I would make is that we investigate why people do not use ACLs. I have been such a person. I haven't done that investigation but I want to, we currently provide security features that nobody uses, and I would like to know why.

Peter Landrock: My approach is the following. Somebody phones us up and says, hey we want to buy software for digital signature and what-have-you, and then I say, OK what is it you want to protect and then they tell me that, and then I say, of course I'll sell, and you can buy, I'll make some money but you'll make no use of it. First we have to understand your whole system, we have to build a model for your system. OK, lets say it's electronic commerce, then we first build a model that reflects all the interesting aspects of electronic commerce in what they want, or maybe it's communication between various governmental bodies in Norway, and when we have understood that then we start adding on the security and that way you can really get into realisations and problems that you would never have thought of if you start with the security.

Wembo Mao: Regarding the notion that trust is something you want to be done that you do not understand. I guess maybe there are two senses, slightly different. One is this, trust on someone, experts, that they can do things properly. The other thing is trust that this whole thing has been done properly without regarding whether they are proper experts or not. If you talk to a committee of experts and you find although these experts are well trustworthy they are not experts and you find people reluctant to use these systems, it's a different sense of trust.

Raphael Yahalom: I wanted to refer to Joan's market failure comment. We're asking, why isn't there a demand for what we're providing, and we just have to reflect back on the last two days of discussion and realise that we don't agree among ourselves. What exactly do we mean by trust and delegation, and how can we expect people who are not thinking about that as part of their professional life to really understand not only what these issues are but what are the implications. But the fact that so far we have not been effective in communicating the concerns does not at all mean that they are not real concerns of significant importance. I want to mention one name as an example, Nick Leeson at Barings. Barings, if they had had a good delegation, trust, whatever you want to call it, system would still be around today because Nick Leeson wouldn't have been able to abuse the delegation and trust, which was inclusively provided to him without any sort of controls within the system. And so the fact that they didn't purchase PolicyMaker or any other favourite off-the-shelf trust and management system does not imply they didn't need it. I think the facts speak for themselves. So it's a question of communicating more and addressing the issues more than asking whether they really exist.

William Harbison: In many real-world implementations we see the process and the control mechanisms for the process being totally separate. The control mechanism is autonomous, constructed in a different way, although with knowledge of the process and touching in various parts. It seems to me that we don't always make this distinction, we try to do the process and the control with the same mechanisms. And perhaps we're missing out on really understanding the problem, and confusing the issues that identify the problem, with solving it, all within the same mechanism.

Joan Feigenbaum: Separate mechanism from policy, that's what we say.

Roger Needham: I think Rafi's point about Nick Leeson is interesting because while there is no doubt that the sad and spectacular tale of Mr Leeson represented a failure of something it's not usually represented as being a failure of what we're experts in and Rafi's point was that that is exactly what it was.

Stewart Lee: However, it was not necessarily computer related. The failure of trust and the failure of delegation was between people not between computers.

Joan Feigenbaum: Computers were more than incidental, they were facilitating. There wasn't end-to-end failure of trust between people but there was a delegation chain that involved a lot of computers, and our good mechanism in there would have stopped that chain before the end.

Bruce Christianson: An empirical observation. Insert a computer into a delegation chain, and people will accept that chain whereas if the chain only involved people an auditor would not permit you to have a chain like that. But somehow people think, well if there's a computer in there and it's doing what it's supposed to do then this is OK.

Stewart Lee: There was a bigger failure of trust in the 1700s in the London Venture Capital Market called the South Sea Bubble. And perhaps the problem is not the speed with which things are done but the way delegations are done.

Roger Needham: There is an interesting point to do with payment systems. One tends to take as just part of the given the ordinary way of doing business, which involves purchase orders, invoices, delivery notes and cheques with all these odd things written on them, for deposit only and account payee only and all of this sort of apparatus. You take this as part of the given, and in fact it's the result of about half a century of experimental development, large-scale fraud, bankruptcy and disaster. These experiments were not meant to be experiments, but it seems to me to be optimistic to suppose that we can move to what's logically a rather different system of doing business and get it right first time. And I've always said a lot of money is going to be made out of electronic commerce, some of it will be made by crooks, a lot of money will be lost as well and this is the process of not being able to get something right first time, it's not something one should feel terribly miserable about.

William Harbison: Could I just address the issue whether we've got our own disasters or not. We do and it's very public, it's called the Internet. I was reading a report about the number of attacks that had been made for access to the DoD, and it was something of the order of a quarter of a million of which I was surprised to read that some many tens of thousands were very successful in their ability to penetrate the system. I almost the next day heard about the trial of the young boy who in Baltimore, I think, who had hacked in unknowingly into an air traffic control system and almost caused a major disaster. More and more of the closed proprietary systems that we have been used to looking at, that we think we know a little bit about, are going onto the Internet because of the prevalence of development tools for IP. There are hospitals which are going to do this and I think we already are seeing a large number of disasters, if we actually look through the statistics of what's happening on the Internet, it's just that none of them have perhaps been quite as massive as the TWA plane crash, but I think we're going to get there quite quickly.

Roger Needham: Well you should be welcoming this, shouldn't you.

William Harbison: And to some extent it's a good test bench for us. I mean in Roger's terms the Internet is a nice open system we can play with and experiment with.

Angelos Keromytis: The classified area of the DoD used the so-called sneaker net or floppy net. If you want to move classified information you put it on floppies and deliver it by hand. Essentially the argument is that if you have very highly sensitive information you do not connect your system to the Internet.

Francesco Stajano: I was surprised that in this workshop on trust this book was never mentioned, this thing was made in our Cambridge group. There are several authors here, perhaps they want to say something about this.

Michael Roe: I could probably say something, I'm one of the primary motivators behind that. It's a bit embarrassing to read out the title because I don't like it, and here it's going to be particularly embarrassing, it's called the Global Trust Register. There now everybody can laugh and throw paper darts at me.

Bruce Christianson: That's like Holy Roman Empire.

Michael Roe: This is a book full of public keys, many of which are the public keys of certification authorities and why this exists is because there is no global root, there are several independent CAs who believe themselves to be the root and will never cross-certify, and so you get a problem with how you acquire those keys, or at least how you get a check on it. And this book merely lists those keys, it notarises them, it's physical paper, it's in libraries, it's very hard for us to change what this says now, it's very hard for anybody else to change it, the CAs whose keys are listed in here have had an opportunity to examine it and to object to it if they thought it really wasn't the value of their key.

Bruce Christianson: And it's got a very strong binding on the spine.

Roger Needham: And it also as I recall has notes as to how good the evidence was.

Bruce Christianson: Well it has notes as to what the evidence was.

Michael Roe: What the evidence was, it ranges from A which was fairly extremely strong, verification of who the person who gave us the key is, what their position in the organisation was and their right to speak on behalf of that organisation, down to D which is we looked in Netscape and it has this key associated with this name and who knows, but here's its value just in case you're interested.

Stewart Lee: And it has an extremely strong disclaimer, as one would expect, if it wasn't there you wouldn't trust it.

Michael Roe: Indeed so, it's not attempting to acquire any liability whatsoever, merely notarising them for anybody who is interested.

Bruce Christianson: It is worth pointing out, which the introduction to the book doesn't make quite clear, that the job of determining how good the evidence is, is that of the reader not the writer.

Michael Roe: Yes, the book describes what we did, you can choose to believe what we did was adequate or not, and it's your problem.

Roger Needham: Personally I hope that the disclaimer was good as I have a substantial shareholding in the company that publishes it. The plan is that subsequent editions will be published by the MIT press.

Michael Roe: This current edition is published by Northgate Consultants, which for those people who don't know is actually Ross Anderson. You can find out how to get a physical copy by looking at
`http://www.cl.cam.ac.uk/Research/Security/Trust-Register` .

Bruce Christianson: And there's a special secret way of knowing whether you've got a genuine copy or not.

Michael Roe: It is available on the Internet in electronic form but you should be aware that downloading the machine-readable off the Internet and looking at the bits lacks some of the physical providence that getting the physical book has.

Roger Needham: The purpose of producing the book was (a) to be useful and (b) to make a point. I don't think it will make very much money.

Michael Roe: The second issue was to set a precedent to deal with UK proposed CA legislation. If the government is going to decree that only appropriately government-licensed organisations can certify people's keys, and furthermore they will only be allowed to do this on the condition that they previously acquired the private key of the person they are certifying, then it is useful to have a precedent of publication in paper which is protected, as a somewhat more established collection of rights that you are allowed to do than the electronic form, saying here are these people's keys. Clearly we're allowed to do this in paper.

Roger Needham: Right well I think this is a good point to draw the proceedings to a close.

Index of Contributors

Francesco Bergadano 95,100
Matt Blaze † 59
Bruce Christianson 154,158,227
Bruno Crispo 118,125
Pancrazio De Mauro † 95
Carl Ellison 36,41
Joan Feigenbaum 45,51,59,227
Simon N. Foley 24,28
Serban I. Gavrila † 67
Dieter Gollman 213,220
Virgil D. Gligor 67,75,177,227
William S. Harbison 108
Angelos D. Keromytis 59
Peter Landrock 137,146,177
Stewart Lee 227
Mark Lomas 131,177
James A. Malcolm † 154
Wembo Mao 102
Roger M. Needham 227
Lawrence C. Paulson 1,13
Michael Roe 168
Ryoichi Sasaki 83,90
Kazuo Takaragi † 83
Raphael Yahalom 177,193,203
Hiroshi Yoshiura † 83
John Warne 177
David Wheleer 105