Lecture Notes in Computer Science 1796

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

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

Bruce Christianson Bruno Crispo
James A. Malcolm Michael Roe (Eds.)

Security Protocols

7th International Workshop
Cambridge, UK, April 19-21, 1999
Proceedings

Series Editors

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

Volume Editors

Bruce Christianson
James A. Malcolm
University of Hatfield/Herfordshire
Computer Science Department
Hatfield AL10 9AB, England
E-mail:{B.Christianson/J.A.Malcolm}@herts.ac.uk

Bruno Crispo
CSELT SpA - Gruppo Telecom Italia
Via Reiss Romoli 274, 10100 Torino, Italy
E-mail: bruno.crispo@cl.cam.ac.uk

Michael Roe
Microsoft Research
St. George House, 1 Guildhall Street
Cambridge CB2 3NH, England
E-mail: mroe@mircosoft.com

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Security protocols : 7th international workshop, Cambridge, UK, April 95 - 21, 1999 ; proceedings / Bruce Christianson ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1796)
ISBN 3-540-67381-4

CR Subject Classification (1991): E.3, F.2.1-2, C.2, K.6.5, J.1

ISSN 0302-9743
ISBN 3-540-67381-4 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN: 10720107 06/3142 5 4 3 2 1 0

Preface

Another year, another workshop. Here are the proceedings of the seventh Cambridge International Workshop on Security Protocols. All very well, you may think, but can there really still be anything genuinely new to say? Is it not just the same old things a tiny bit better?

Well, perhaps surprisingly, this year we discovered some radically new things beginning to happen. The reasons in retrospect are not far to seek: advances in technology, changes in the system context, and new types of consumer devices and applications have combined to expose new security requirements. This has led not only to new protocols and models, but also to known protocols being deployed in delicate new ways, with previous fragilities of watermarking and mutual authentication, for example, becoming desirable features. At the workshop we identified several of these developments and began to map out some lines of enquiry.

This volume brings you a selection of deliberately disputatious position papers, followed by not-quite-verbatim transcripts of the discussions which they provoked. As always, our purpose in making these proceedings available to you is the hope that they will move your thinking in an unexpected direction. If you find your attention caught by something here, if it makes you pause to reflect, or to think "why, that is just *so* wrong", then good. We're waiting for your mail.

Thanks to Stewart Lee and the University of Cambridge Centre for Communications Systems Research, for acting as hosts for the workshop, and to Roger Needham and Microsoft Research Limited (Cambridge), for providing us with the use of their meeting room and coffee machine.

Thanks also to Dorian Addison of CCSR and to Angela Leeke and Margaret Nicell of MSRL for keeping us organized, and our especial gratitude to Lori Klimaszewska of the University of Cambridge Computing Service for her Promethean transcription of the audio tapes, which this year featured "echo kickback as well as the usual distorted sound".

Finally, each of us takes full responsibility for the accuracy and completeness of the material contained in these proceedings. Errors and omissions, on the other hand, are the responsibility of the other three fellows.

February 2000

Michael Roe
Bruce Christianson
Bruno Crispo
James Malcolm

Introductory Remarks

Michael Roe: We always try and have a theme and then people submit papers. The usual thing that happens when you're running a conference is that people resolutely submit a paper on the research they're doing that year, regardless of whether it's got anything to do with the theme or not.

What I thought was going to be the theme — because I'd been deeply buried in it for the previous year — was entertainment industry protocols. We're seeing a great shift in the use of the Internet from the kind of educational, military, and industrial use of networking towards home users buying entertainment, and this causes you to completely redesign protocols. I've been particularly looking at copy protection and digital watermarking, and when you look at those kinds of protocols you discover they're just not like the usual authentication and crypto key exchange we've been trying to do for the last fifteen years. They're fundamentally different, even the kind of asymmetric properties you want in a digital watermark — that anybody can verify it and see that this is copyrighted data and nobody knows how to change it — you think, oh that looks like public key cryptography, but then you discover, no it's not public key cryptography, it's something that's similarly asymmetric but it's different.

And so I thought a theme for the workshop could be how these completely new application areas cause us to come up with protocols that are completely different from what we previously discussed. But from people's submissions this didn't look to be what everybody else was doing.

The second theme that Bruce suggested was auditability of protocols: What do we need in a protocol in order to audit it? What does it mean to have audit support in a protocol? Bruce, it was your idea ...

Bruce Christianson: Well when something goes wrong, sometimes the question is, what actually happened? And then you work out what state you ought to be in. But I think we know that in the protocol world there's often not a fact of the matter about what actually happened. So we need instead to have some way of agreeing a story we're all going to stick to, about what we're prepared to say happened. Some kind of integrity property for restoring the state. It seems to me that several different pieces of work have each got one corner of that particular problem, and it might be interesting to discuss some of the relationships between those approaches that aren't usually discussed.

Michael Roe: So it's going to be a mixed bag this workshop. A general overall theme still might be the changing environment and the changing application areas, it's just things have changed so much and become so diverse that they exceed my ability to predict what the papers are going to be about.

Une Mise en Thème

an exchange of e-mail

stardate February 1999

From: B.Christianson@herts.ac.uk Fri 3 February 1999 14:45
To: E.S.Lee@ccsr.cam.ac.uk, m.roe@ccsr.cam.ac.uk
Subject: protocols workshop

there is some discontent with the present ‘theme’:

> one of the strengths of the earlier workshops was that they didn’t
> concentrate upon a particular application area and so encouraged
> people with different interests to come together.

how do we feel about ‘making protocols auditable’ as an alternative?

this theme includes the issues of how a protocol may be audited against more than one policy, what audit records are required and how they are to be kept, the nature of the criteria for success etc.

bruce

===

From: Prof E Stewart Lee <E.S.Lee@ccsr.cam.ac.uk> Fri Feb 5 1999 15:08
To: B.Christianson@herts.ac.uk, m.roe@ccsr.cam.ac.uk
Subject: Security Protocols Workshop

> this theme includes the issues of how a protocol may be audited against
> more than one policy

This is unwise. Generally, a protocol can enforce more than one policy iff one the policies is a subset of the other. This is a well-known result of composition theory. Policies are even more difficult to compose than objects. Two policies that do not satisfy the subset criterion can sometimes be enforced iff there is a requirement that one be enforced before the other -- e.g.: Mandatory Access Control before Discretionary Access Control. For another example, Bell LaPadula is a policy. So is Non-interference. Their composition is extremely difficult (because NI doesn’t have DAC, amongst other more technical reasons).

Stew

===

From B.Christianson@herts.ac.uk Fri Feb 5 1999 15:06
To: E.S.Lee@ccsr.cam.ac.uk, Michael.Roe@ccsr.cam.ac.uk
Subject: security protocols workshop

i think you are the first to submit a position paper, stew ;-}.

i'm not welded to those particular issues, but the better to inflame debate:

policy A: (laundry) no payment implies no laundry
policy B: (customer) no laundry implies no payment

neither policy entails the other.

bruce

===

From Michael.Roe@ccsr.cam.ac.uk Fri Feb 5 1999 15:23
To: Prof E Stewart Lee <E.S.Lee@ccsr.cam.ac.uk>
Subject: Re: Security Protocols Workshop

Hmmm ... we're starting to jump into the discussion that should be held *at* the workshop here.

It is a fact of life that many protocols involve communication between parties who have conflicting interests. Each party, if given sole authority to set policy, would define a policy which conflicts with that of the other party. Reconciling these conflicts *is* an imporant issue in protocol design, even though we know that there is no *mathematical* tool which will cause real conflicts of interest to magically vanish.

I vote we keep Bruce's original sentence.

Mike

Table of Contents

Keynote Address: The Changing Environment
(Transcript of Discussion)

Roger Needham

Microsoft Research Ltd.

What I wanted to spend a few minutes talking about is changes it the environment and purpose in which protocols are conducted. When Mike Schroder and I wrote our paper in 1977, published in 1978, one of the things we did was to be quite explicit about what we assumed to do with the environment. We assumed that we had principals, whether human or mechanical, that wanted to establish a secure connection, and the assumption was that the principals themselves worked in the way they were supposed to work. If they were people they were honest people, if they were programs they were correct programs. But the environment was thoroughly wicked and everything about the environment conducted itself so as to cause the whole procedure to fail if it possibly could.

That was one assumption. There are a number of other assumptions to do with the environment. For example, that communication was slow, that encryption and decryption was slow, and therefore it was desirable to do them as little as you could; that there were no reliable source of time and therefore that time should not be used whatsoever; and that state was evil. Obviously the principals trying to communicate should set up state, that's what they were trying to do, set up shared state. But servers that help them along should not maintain any state — partly because for reliability, you would have to replicate them and keep them safe and stable in a number of different places, and partly because it would involve things like disc transfers at an embarrassing time, and these were inordinately slow.

Another assumption was that there was no way in which people could have any at all extensive repository of secrets, because the kind of replaceable disc which was released at that time, they were about 2 megabytes capacity and they were about this diameter. They were certainly not something you could put in your pocket or your wallet or your handbag. The other assumption was that computers, although they might be used by one person at a time, they were not personal in the sense that they were owned by an individual. If your machine didn't work, you wandered down the hall until you found somebody who was away, and used the one in that office. They were a serially reusable resource.

Now, it must be obvious that practically every one of the assumptions I have listed isn't true anymore. We do have a reliable source of time; it is possible to maintain personal secrets without it being terribly clumsy; you could if you wanted have a stateful server — non-volatile RAM had not been invented at the time we were writing.

And finally, the assumption that the principals engaging in a communication were persons or things of goodwill and that the rest of the world was evil isn't true either anymore. This is particularly true of electronic commerce, where the

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 1–5, 2000.

participants in a secure integral transaction may very well have cheating each other as a major goal and not be particularly worried about interference from outside. I would claim that this collection of changes in assumptions adds up to really taking a quite different view from what we do. And I suspect that something that has happened is, that since the study of security of protocols became a cottage industry in the 1980s, people have let the environmental assumptions of the time just become part of the way they think.

It is because the subject has become, in some slightly bad sense, academic. It's practised by a lot of people who actually don't know much about security requirements in the real world, but they can play table-tennis with the symbols and become very skilled at it. So I think it's worth considering — but I have not done this consideration myself — whether we need to revise any of our attitudes.

Another thing that was assumed was that communication was pairwise between an enormous number of principals, one may think of the number of people. This is not to the point either in very many contexts. In many contexts you have organisations which have got an internal network like Microsoft does, which is not particularly secure, it's simply separated. You have communications between people in Microsoft and people in other organisations, say Compaq, where if you wanted to be secure about it, the bit of communication that needs to be secured is the bit that goes from the firewall at the edge of Microsoft to the firewall at the edge of Compaq, and that's going through the wicked world. Now what that has done is to decrease the number of pairs of things that need to have secure connections between them enormously, and it's arguable that this makes possible a completely different approach to doing it. For example, you might say that Microsoft and Compaq equip themselves with a suitably large amount of one-time pad, and every time a visitor physically goes from one place to the other, the visitor carries a few gigabytes of one-time pad in his or her briefcase, and hands it over and it all gets refreshed, and there isn't any infrastructure.

I don't know whether that is particularly sensible, but it's feasible, it's the sort of thing that just wouldn't have been feasible before, and it's worth at any rate thinking about. Similarly, people go on about public key infrastructure. When I used to run the Computer Lab, I thought about what it would be like for us to do our business with our suppliers by electronic commerce. I found that the Lab had about 1800 suppliers which it added to at the rate of one every couple of weeks. We added a customer at the rate of one every five or six weeks. Now these are small numbers, the Computer Lab is not particularly a small organisation and it would be perfectly feasible to make arrangements between it and any of its business correspondents on a pairwise basis, for the Head of Department to write to the company saying we would like to establish an account with you and we will be bound by any electronic signature which is verified by the following public key, puts a stamp on it and sticks it in post, done, no public key infrastructure needed, and you can in fact bootstrap from that up to all sorts of other transactions.

I have not carried any of these thoughts through but I do think that enough has changed, and enough is changing, for it to be appropriate to take a fresh

look at what we do and why we do it, and the circumstances in which we expect it to work, and I shall leave it at that.

Matt Blaze: I'd like to quibble with two things you said, in agreeing with your premise that we need to think about whether the model that we evaluate protocols against is still valid. The first is the virtual private network model that you mentioned, the idea that Microsoft and Compaq talk to each other, so they protect the edges of their network and simplify the problem in that way. Is it actually true that the Microsoft and Compaq networks are more secure against attacks against Microsoft and Compaq than the Internet at large is? I wonder whether the focus on an external threat, which we traditionally looked at in evaluating network security, might be rather foolish. The internal threat may in fact be much greater.

Reply: I think you are very likely right. If you want a really cynical response, it is that corporations behave as if their intranet is secure, and therefore they would feel comfortable in an environment where it was protecting between one corporation and another. Whether they ought to feel comfortable is an entirely different matter, but in practise they do.

Matt Blaze: Because I think internal security will be with us as a requirement for a while. I'd also like to raise the question about the notion of electronic commerce. For a long time we as security people have been talking about the virtues of converting to a paperless workflow system with digital signatures and so on, we've never actively succeeded, but all of a sudden, at least in the United States, we're now seeing laws. States are rushing and competing with each other to see who can be first to pass laws recognising digital signatures as being equivalent to handwritten signatures. And now I think it's gotten away from us. Certainly digital signatures do not have all the same properties as handwritten signatures, and I think it's a tremendous mistake to design security protocols that simply replace the handwritten operations with digital ones, when they're not exactly analagous. In particular, I can't think of an easy way for somebody to steal the ability for me to write my handwritten signature, but I can think of many easy ways they do that with my digital signature.

Reply: I think that's for sure, and my expectation is that, as electronic commerce begins to happen on a big scale, there are going to be some spectacular and ingenious frauds. The present system of doing business with purchase orders and invoices, and copies of the purchase order and cheques and things like that, didn't come into existence overnight, it came into existence over quite a lot of years, with a fair amount of experience of fraud along the way, and I expect the same thing to happen in the electronic world. I think you are absolutely right because people will assume that digital signatures are isomorphic to paper signatures, and that will generate a certain amount of confusion.

Matt Blaze: Some people look forward to it.

Reply: I think some people look forward to it.

Ross Anderson: It may be even worse than that because various things aren't always given the presumption of validity that digital signatures are. This rule erodes the traditional consumer protection that we have in the UK. We have,

for example, credit card legislation similar to your regulation in the States which says that if something goes wrong between about £100 and £3000, then that's the bank's problem not the customer's. If you replace that with a law that says that all digital signatures that are presumed to be valid, then suddenly all the risks are dumped on the poor customer. And so you can expect lots of frauds will follow because of the moral hazard. We think that this is one of the big risks of digital signatures. It's actually coming about because digital signatures have been used as an excuse for a sleight of hand which transfers power from the individuals to the state instead. You don't hear about this at Crypto conferences but perhaps it's an awful lot more important than any of the technical aspects.

Reply: Yes, and it is for reasons connected with that, that when I was saying that something like the Computer Lab might engage in bilateral negotiations with suppliers, I should perhaps have added that I wouldn't expect the Computer Lab to use the same public key for all its suppliers, it might easily have one each. That in itself would be an anti-fraud mechanism, because if somebody steals the signing key they might put themselves in a position to buy a large quantity of toilet paper at the Computer Lab's expense but not to do anything else.

Larry Paulson: I made this remark last year so I'm sorry but I know that in the real world there are signature stamps that are used to sign cheques and there must be controls on those.

Bruce Christianson: Usually they're locked in the safe, but the tremendous advantage is that if somebody's stolen it you can tell — you look in the safe and it's gone.

Stewart Lee: To reinforce what Matt said, somebody — I've forgotten who, it's just fallen out of my mind, it's what happens when you get old – this consultant in New York did a survey of incidents of fraud that involved computers, for security incidents, it involved computers, and discovered that 68% of them were insiders, Bob Courtney, that's his name, and I see no reason at all to expect that the Internet will make any difference in the proportion, about two thirds of them will done by insiders. To reinforce another thing that Matt said, I'm on the Matt bandwagon today, you mentioned that you foresaw that the parts of traffic on the Internet that have to remain secure, could be decreased dramatically if it's intranets talking to intranets sort of thing, but I feel that the Internet is already at a place where there is an enormous number of credit card numbers and expiry dates going over it, and I am very loathe to do that myself because I know how easy it is to observe what goes on, and it is very difficult in fact as a consumer to recover from that. If I order something from some company in Texas and it gets sent to me as information by e-mail, for fifty quid or something like that, it's very difficult to get my fifty quid back, if it's actually somebody else's. How do I prove it?

Reply: Yes, indeed, and what I suspect are the patterns of doing business may change.

David Wheeler: I would speculate that if two thirds of computer frauds are insiders, then two thirds of any fraud are insiders, though it's very easily hushed up.

Stewart Lee: The Bob Courtney survey showed that the small thefts were always publicised, it was the big ones that were hushed up.

Virgil Gligor: Actually, one of the things that Bob showed is that it was more than theft and computer crime. Accidents and mishaps, acts of God, floods, fire, air conditioners blowing up, accounted for most of the losses. Fraud altogether, out of this whole thing, was only about 20% to 23%.

David Wheeler: That's *discovered* fraud.

Virgil Gligor: One thing that bothers me, one difficulty with the Internet. Maybe I'm the only one who's bothered by this. We are locatable at fixed places, we're fixed entities, we are given some sort of Internet address, perhaps two or three, but those are fixed. So we are sitting ducks for people who are willing to attack us. In some sense this happens in reality, we have one place of residence, but generally most of us don't hold much of our valuables in our homes anymore, so we spread our funds in multiple areas, do all sorts of protective things that we learn how to do, and in the Internet we aren't quite so able to do so. In other words, if we were to spread out our wealth of files and data across the Internet somehow, I would feel much better protected. Which reminds me of the paper that Yves Deswarte wrote some years ago in Oakland about fragmenting files and sending them all over the place. Well here is the Internet, we should be able to do that, and we should be able to do a lot more, yet we are not doing it. Does that make any sense to you, is this valuable, is it just a flight of fancy?

Ross Anderson: I think that the Internet threats are wildly overstated by people who are trying to sell the security software. Until a few years ago banks, VISA and so on, were trying to tell everybody: don't do credit card transactions on the net, wait for SET. Now they've got some past history on their books they've changed their mind, and thankfully for them people are beginning to recover from the initial attack of panic. Now I only monitored what was going on closely until last September or so when I handed over the job of reading the relevant banking technology articles to somebody else in the context of Security Reviews, but until then there had been no case anywhere of somebody taking a credit card number off the net. If there had been a case it would have been reported, it would have been all over the headlines. But there was one case where somebody hacked into a merchant server and got credit card numbers that the merchant should not under his merchant agreement ever take. Actually hacking stuff off the net is hard work. We all assume in theory that it can be done but it's not in practice how you would go about getting credit card numbers.

Composing Security Properties
(Transcript of Discussion)

Stewart Lee

Centre for Communications Systems Research
University of Cambridge

What I'm talking about is simultaneously enforcing more than one security policy. What I'm really talking about is when it's over a network where you have two machines that are connected, and you have a security policy in each end. I thought it would be useful to try and decide what we mean by a security policy. It's a directive on the goals for access and integrity and other things, it's an objective or a goal. What I'm going to do in this talk is to very quickly review a number of known security policies. I picked on ones that have been published, that are reasonably well known. I am then going to demonstrate that these policies are inconsistent and any attempt to enforce both of them at the same time is doomed.

The Orange Book is one of the obvious ones. We all like to shoot at it, but there it is, it's a policy and it was embodied in a tract that was first issued in 1983. I very consciously call it a tract because some sort of religious fervour is associated with it. It was followed by a number of other criteria: the British produced the green book, the Canadians produced the blue book, this is brown, the common criteria. That is, the original document that describes it was brown and one must be consistent if nothing else. The Orange Book policy has six fundamental aspects to it: continuous employment of protection; objects have access control labels; individual subjects must be identified. The orange book governs the access to objects by subjects. Objects are passive subjects of actions. People have to be responsible for their actions; it's practical to evaluate the system, in other words it isn't a spaghetti junction of hand written code, and continuous protection must be involved from concept right through to installation.

The Orange Book is modelled on something called the Bell-LaPadula model: reading data at a lower level is allowed, writing data to a higher level is prohibited, destructive writing. You can always write up if it's sending information up but you can't destroy something up there.

Another policy, this time an integrity policy, is a very old paper by Biba. The subject's integrity security level is affected by the integrity security level of the object that it observes. If a subject observes a low integrity object it becomes a low integrity subject, and so on.

Biba has fallen into disuse but it's a policy. Clark-Wilson is a policy that is quite often used. Ross Anderson has used a version of it in medical data. You have unconstrained data items and constrained data items. This is a transaction processing policy for the commercial world and it includes the N-man rule; an event journal, which is where auditing comes in; an integrity verification procedure which was shrouded in mystery in the original paper and still is shrouded

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 6–14, 2000.

in mystery, and so on. Notice here the terminology is different than in the previous two policies. Not only is the terminology different but this is a policy on a transaction processor that runs on some computer system that has an underlying operating system. Both the previous two have really talked about operating systems.

And finally information flow policies. Information flow policies are governed by some sort of process where there's things with high security level and low security level coming in and things of a high security level and low security level going out. There are two of them that are commonly referred to, one of them is so-called generalised non-interference and it says that the low out cannot be interfered with by either the high in or the high out so that you cannot tell from the low out whether there was any high in or high out there or not. It's not interfered with.

The other one is subtly different. It says nondeducibility and you have exactly the same arrangement but now an observer examining low out cannot deduce with certainty, anything about high in or high out. The key thing here is with certainty, cannot be deduced with certainty. So that something is nondeducibility secure if there is one chance in ten to the something that the message doesn't say what in effect it does say. Non-deducibility security is frequently used in circumstances that involve cryptography because if the process that I spoke about is cryptography then clearly the high in affects the low out.

There are problems when there are several policies and I've got five issues there: information flow versus the Orange Book; Clark-Wilson versus the Orange Book; the Orange Book versus the Orange Book and so on. All of these things have problems of varying difficulty. Information flow really knows only about read up and write down, it doesn't know about what the Orange Book calls discretionary access control at all. The Orange Book has about five other policy things that are just not contained in information flow, so that if one side of the conversation is running information flow and the other is running Orange Book, it won't pass.

Clark-Wilson has the same problem. You have this terminology of constrained data items and unconstrained data items and it doesn't know about a security level. So you have terminology problems. Also, Orange Book doesn't know about N-man rule or UDIs or an integrity verification procedure, and so on.

This is the classic thing where one system has top secret and secret in it, the other system has secret and confidential in it, both are approved for those circumstances, neither of them are approved for the combination of three pieces of data and that is a pretty standard problem.

Orange Book, when doing subject to subject, must be interpreted, because Orange Book really applies to subjects accessing objects. You have to do an interpretation of it, and every time you do an interpretation you're dealing with your own local version of some policy, and that local version may well be inconsistent with others. For instance, when you're dealing with subjects accessing subjects you have flow of control which is an issue. How do you deal with parameters? Is that reading or writing? How do you deal with results? Is that reading or

writing? What do you do about exception handling? These are all places where some things can get into trouble. Sometimes there are other problems. Very few policies other than Clark-Wilson are covered, and it means that one system may need enforcement of something about which the other is ignorant, and that's just not going to work.

Now, something that has to do with inconsistencies. Just to bring us back to earth here, just UNIX against UNIX. It's really not a policy but a mechanism. There are many brands of UNIX and not all of them define the access permissions quite the same way. For instance, the concept of ownership was a little bit different in System V UNIX as against POSIX as against BSD. I presume it's different again in the more modern versions of UNIX. Consequently you have to be very careful of what you do. So what we're trying to do is called composition, we're trying to compose out of several components a system where we can predict the security policy that the system will enforce by knowing the security policies or properties that the components will enforce. There's no known solution to the general composition problem. There is a solution for the general composition problem where the two policies are information flow policies, that is known and was published about a year ago[1]. It's known that information flow policies can be composed and you can predict the resulting security policy of the composed system, from the security policy of its components. But that's the only case and it's probably the only case that is practical and considered because the other cases are not sufficiently well analytically modelled to achieve anything.

So what I'm saying here is that two policies cannot be simultaneously enforced unless they are essentially the same policy; or one is a subset of the other and you can use the stronger rather than the weaker; or it's OK to enforce neither of them well; or one of them not well. And this is the point that I wanted to make, and it pertains to this business of auditing the process of a protocol. If you're just auditing for the sake of auditing, keeping a journal of events for the sake of keeping a journal of events, and then auditing that journal of events is busy work involving a great deal of magnetic storage and so on unless you have some objective for doing it. There are a few objectives for doing it, like pinning your finger on who initiated this nonsense, which is reasonable to do, but trying to use this to ensure that some security policies of the end users are being enforced, is unlikely, in the general case at least, to succeed.

I was all inspired by Bruce Christianson. I apologise for the review of well known stuff that was contained in there, but there it was.

Virgil Gligor: Did I understand correctly that you suggest that we use even non composable policies but we audit activities?

Reply: I didn't suggest it, Bruce suggested it.

Bruce Christianson: Stewart initially made an unguarded assertion, which was that you can't enforce two policies simultaneously unless one is a subset of the other. I said, well suppose I have a policy that says you don't get the goods

[1] A. Zakinthinos and E. Stewart Lee, "A Generalised Theory of Security Properties", 1997 Symposium on Security and Privacy, Oakland, California

unless you pay for them, and you have a policy that says you don't pay for the goods unless you get them. Neither of those policies is a subset of the other.

Virgil Gligor: Ok, but it could be said that they were not inconsistent, so you can enforce both of them simultaneosuly.

Bruce Christianson: So then you imagine enforcing them at the same time. The situation will often end as: I want to look at the audit trail to see whether this policy is being enforced, or has in fact been complied with. You want to have the sort of audit trail where you can do that, possibly with a specific policy that you didn't have in mind when you kept the trail. You have the general form of policy but you didn't have that specific policy.

Virgil Gligor: So is it the case that I can take two provably incompatible policies and use them, but because I audit I can discover whether the incompatibilites are exercised to commit fraud?

Bruce Christianson: That's one of the arguments.

Stewart Lee: I'm not at all sure I agree with that.

Bruce Christianson: A little point of difference between Stewart and myself.

Stewart Lee: I believe that a journal of events can display what happened in the communications but it does not display what happens in the end systems. It may well be the case that sending something is legal in certain circumstances and prohibited in others. How is the analysis of the communications going to display which of those is in fact true, without also keeping a journal of the things that went on in the two end systems?

Bruce Christianson: Sure, it's clear that an audit trail of a protocol involves more than just keeping a record of messages which were sent and received. You've got to keep track of internal state, or at least some projection of it.

Stewart Lee: May I point out that Bruce and I are using different words for precisely the same thing. I keep talking about a journal of events and he keeps talking about an audit trail — he's more American than I am.

Ross Anderson: I'm not sure I agree with your claim that Bell-LaPadula and Clark Wilson are inconsistent. To take the standard teaching example that we use with undergraduates, consider an Air Force logistics system, which has two separate databases, one is restricted and one is secret, and there is a pump that pumps up a copy of the restricted database continually to secret. The MoD actually spent five hundred million trying to build one of those things. It is quite clear that these policies can be simultaneously enforced. If you have a look at the medical policy that we did for the BMA, that contains Bell-LaPadula as a strict projection because the seventh rule says that you can copy information from one file to another only if the access control list to the first is contained in the access control list to the second. This gives you an example of how it's possible to generate a security policy that is non-trivial and different from the original Clark-Wilson model, but which contains Bell-LaPadula in a useful way that mirrors actual practice in the real world.

Stewart Lee: You know Bell-LaPadula was not a policy it's a model. Bell-LaPadula is a model of the access tree and access matrix and so on. For those

who haven't looked at it in detail, it's nothing other than simple set theory. It isn't a statement of the goals and objectives that we're attempting to achieve.

Michael Roe: But that's critical, there is a big difference between the human level explanation of what the system is trying to achieve — e.g. people's medical records are confidential and you don't want everybody to be able to read them — and the particular pieces of mathematics that are enforced at the program level in the operating system.

Stewart Lee: The Orange Book doesn't call those policies. If you look at the Orange Book itself, I'll show you.[2]

Michael Roe: I think you're right, these things that we informally call policies, like Bell-LaPadula, are not really. We were really talking at cross purposes when we had the discussion of whether you could compose policies or not. What you were talking about is these things that the Trusted Computing Base enforces, whether you can take two of them and just glue them together. If they use fundamentally different mathematical representations of what's going on, taking these two little pieces of computer code and sticking them together doesn't work. What we were talking about was composition of the human comprehensible policies — If you have two different organisations both with different goals, could you produce a system that met both sets of goals.

Ross Anderson: The answer is very often yes.

Stewart Lee: In special cases it's often, yes, but it's not always yes.

Frank Stajano: If the answer is no, why would they do business with each other? If they're not both satisfying their own policies, why would they ever do business if the answer is not yes?

Ross Anderson: But the battle of the forms is standard commerce. The purchaser and the seller each send their standard forms to each other, each of which says I am right and you are wrong. Whose prevails in court? This is a well-known unsolved problem in the real world that leads to many real disputes. People argue over which was posted first. It's not a thing you can solve with computers, it's a real world problem.

Virgil Gligor: I don't know why we have an expectation that these policies be compatible with each other, when you realise that we don't. I mean in real life we live under different sets of policies, sometimes completely different policies. When you merge two organisations, which happens very often nowadays, it takes a while to adjust. So maybe Bruce's idea is really fundamental in some sense — that we allow incompatible policies to co-exist for a while, we audit what accesses take place and try to weed out the violations for a while, and then adjust it.

Stewart Lee: It's the only way we could behave in your example, as you've got to melt together, to take a random example, as you've got to melt together Microsoft and Sun, they have entirely different corporate policies.

[2] The Orange Book actually says "A statement of intent with regard to control over access to and dissemination of information, to be known as the security policy, must be precisely defined and implemented for each system that is used to process sensitive information."

Virgil Gligor: The ideas that some people put forward nowadays are those of having incompatible policies co-exist, have a meta-policy designed to resolve disputes and have audit to figure out what happened when people exploited the inconsistencies.

Stewart Lee: That would be an extremely interesting research topic, but again the meta-policy is going to have to be principles.

Virgil Gligor: Call it exception processing.

Stewart Lee: Yes, exactly right, but then you've got to detect the exceptions.

Ross Anderson: Sure. This is the point, we can't even define what the penetration is in some cases. For example Microsoft may have a policy that communications between Microsoft UK and Microsoft in Redmond will not be read by any third party, and the intelligence agencies may have a policy that they will read all traffic between Britain and the USA.

Virgil Gligor: I don't necessarily mean penetrations — I mean people taking advantage of inconsistencies. So we have some inconsistencies which will say access denied, and the flag will say access denied, let's see if it's real denial or it's an inconsistency of policy. But there are some other cases where access is allowed due to some inconsistency in the policy, in that case no flag is raised, but there is already an audit trail where we can go back to and find out what went wrong.

Stewart Lee: That's what I had in mind when I wrote down at the bottom of the last slide, it's OK to enforce neither of them perfectly, but if you're going to do that, you're going to have to know in each circumstance what you didn't enforce.

Ross Anderson: So you end up with a demon that can override either or both security policies if it feels like it.

Stewart Lee: Not necessarily, you don't have to do this preventatively, you're merely assigning responsibility.

Virgil Gligor: The philosophy here is very different, there is a deliberate hole in the policies, not in the implementation. There is a hole, it's known, but we also know that we're audited. Audit has a good deterrent effect possibly. We allow the inconsistency in allowing access, which means we are allow the exposure — but we audit it. Secondly if we're denied access because of inconsistency then we have management override by the meta-policy. So you live with policy denials and with holes and we handle both of them in different ways.

Stewart Lee: It would be very interesting to try and take a look at some of these policies, and others, there are others, to see whether there is any practical way that we could detect these circumstances where one or both of them are not being obeyed by looking at their communications.

Ross Anderson: We've built this into the medical policy, we say you can have emergency override but you must tell the patient as soon as practical afterwards, there's a reference been looked at by somebody that he didn't approve of in advance. This is implemented and fielded in the hospital systems, it's im-

plemented, any doctor at the hospital can see any patient's record, but the fact of that that override is written to a file that the patient's own doctor looks at.

That's the half of the problem that pertains to the half of the system that's using the BMA policy, if you like, and if you're also doing Bell-LaPadula because it's a military hospital, then you also have to have a management override for the nurse who looks at the general's record because he's got shot.

Virgil Gligor: The other half is putting together policies with too much denial and not much work done. In that case when you don't have access, it has to be overridden occasionally too by the meta-policies. So I see two parts in the inconsistency, too much denial so you don't do useful work, in which case you have an override, and not enough. If they accept inconsistencies in real life why shouldn't they accept them in computer systems? Why should they always insist that everything be absolutely perfectly composed?

Stewart Lee: Oh I agree, and I believe that as a short term measure, as it were, it is very sensible to be able to examine the communications between two computers that are obeying distinctly different policies, but I wouldn't purposely set up a system that consciously did that.

Virgil Gligor: I'll tell you why I would. I would because there are two things that I noticed the Internet: One is that at any point in time, there is one system in the Internet which is vulnerable to penetration. So we live with vulnerability from the point of penetration resistance and we have to be able to do something about that. The second thing is that at any point in time we have two parties talking to each other, they don't know about their policies or couldn't care less about their policies, because of the dynamics of the global system. Consequently in some sense we have to learn to live with these compositions, perhaps unwanted from a security point of view, but practical compositions of policies. If I were to design a system specialised for a particular application, I would agree with you, I would actually make sure that things are compatible and I don't get too many denials and too many holes, but if you're talking about a loose confederation of policies, just people and organisations talking to each other, then probably we have to live with this in the new world that I just described.

Ross Anderson: The whole thing comes down to monitoring. How in practice you're going to determine whether you're doing too many management overrides or not is whether your managers begin to perceive this as being tiresome and if they do they will try and turn it from management into administration by picking up some automatic rules which will enable it to be subjected to processing.

Matt Blaze: I'm just trying to figure this out. You're managing multiple security policies and using logging and audit as the fallback. Which policy governs the management of the audit of the log?

Stewart Lee: You've hit a crucial point. That's what Virgil has been saying, that you need a meta-policy to do just that and that meta-policy is going to have to acquire the journal of the events that are flowing back and forth.

Matt Blaze: In practice that's just a completely intractable problem.

Stewart Lee: Yes I agree.

Matt Blaze: In almost every system I can think of, the logs are the single most valuable thing to protect, both for integrity and confidentiality.

Bruce Christianson: Sure, but the fundamental question is what do have to put in the log?

Stewart Lee: I do not disagree that it's a very difficult problem. One may develop a need for vast quantities of disc and so on, but in the end it would be nice to know whether it could be done if we were willing to make the commitment. I can't answer that. I'm rather more negative than others here about whether it can be done, but it would be nice to consider what the meta-policy would have to be, given that the end point policies are known.

Matt Blaze: I think one property of that meta-policy if you're keeping vast quantities of logs is the existence of some party who is authorised to create the policy with respect to the log.

Virgil Gligor: Well I think the idea here is a bit different. Since we don't make the policies common, or unify them, perhaps we can come up with some common meta-policy, for example managers looking at the log. So the idea is that basically you minimise the commonality in some sense as much as you can, because that's easier to install and manage.

Matt Blaze: I'll give you as an example. UNIX systems traditionally have logged things like failed login attempts, back in the days when we naively believed having a password was the right way to protect login to the the system. It was discovered early on that the most common login mistake was to get out of phase with typing in your login name and your password, and so what appears in the logs is "incorrect login attempt to <Matt's secret password>". Of course it doesn't log the incorrect password which was `mab`, my login, but would it log the password itself.

Virgil Gligor: There is no attempt here to minimise the complexity of the problem but there is an attempt to say that we live with these things all the time and we've got to find some clever ways to do that in computer systems.

Michael Roe: You have to do this anyway. With things like compartmented-mode workstations that really try and enforce Bell-LaPadula, you discover that the real policy — exactly which bits of information really are sensitive and are not supoosed to be given away — is actually somewhat different from what the TCB thinks. So you need the button to push to say this thing I've got here is not actually classified information even though the Bell-LaPadula model says it might potentially be.

Stewart Lee: It may be unclassified information that has been used in an environment where classified information can seep into it.

Larry Paulson: Maybe I'm naive and don't know much about real-world security, but I try to teach the security course to students. I don't really believe it. I know for example in the Bell-LaPadula model there are no write downs, so when the secretary types up the spy's field notes and sends them to the general, the general is not allowed to send her a thank-you note because she wouldn't be cleared to read it. I can't believe that anyone would implement it in fact, that's why it's hard to be precise about it.

Stewart Lee: What you do is you have your secretary cleared to secret.

Michael Roe: That's right, there needs to be an example of why Bell-LaPadula, if you just follow the mathematics, is completely mad in any real world application. The real policies are, approximately, Bell-LaPadula with overrides where common sense is used and with appropriate authorisation.

Bruce Christianson: The big effect of the lattice models is actually the effect it has on information policy.

Virgil Gligor: Actually the biggest effect of the lattice model is that it breaks applications. This is what killed it everywhere, including the military.

Bruce Christianson: There's another line of thought that this argument produces for us. There's a lot of people who say policies should give you an answer: if you ask your policy whether you're allowed to do something, it will say either yes you are or no you're not. Generally, if it says "I can't make up my mind", you say to yourself either the implementation's not very good or this policy hasn't been thought through. Whereas I think what Stewart's argument shows is that if you've got two policies which are complete in the sense of always giving you an answer, and they're not the same, and you try and stick them together, you are really in trouble. In real world policies where we've said, "Look at this lazy, feckless policy classifier, he seems to have gone out of his way to avoid giving a clear answer about what happens in certain circumstances", I think in a lot of cases the correct response is not to say "For heavens sake why can't they just go away and do it properly?", but to say, "OK, what other policies are they going to try and compose this with, and is that why they're being so vague about this, based on their practical experience from doing this in the past".

Auditing against Multiple Policies
(Transcript of Discussion)

Mark Lomas

Goldman Sachs International

I should preface this by saying that I'm not actually going to give you a solution to anything, I'm just going to explain why auditing against multiple policies is actually useful. Well it's both irritating and useful. It's a necessity for some people.

But first I'd like to make a comment on something that Roger said earlier. Roger mentioned that there are a number of underlying assumptions that were made when he and Mike Schroeder were writing their paper on authentication way back in 1977, I think it was published in 1978. One of those assumptions was that it's a good idea for servers to be stateless. In the environment where they were working that was possibly a reasonable thing to decide, unfortunately, an awful lot of people since have copied that example and decided that it's inherent in authentication services that they ought to be stateless.

Now this actually leads to a problem for people who want to do auditing. In essence, the problem with the Needham-Schroeder protocol – at least one of the problems with the Needham-Schroeder protocol – is that the authentication service itself never actually gets a guarantee that what it was participating in actually succeeded, and therefore you can't make the audit trail. Now you can't blame Needham-Schroeder for the faults of their successors, but one of the successors of the Needham-Schroeder protocol is Kerberos, which is widely fielded – I have to admit that we use it within our organisation. Because it was following that example, it is essentially unable to create sufficient accurate information in the audit trail for us to comply with what I would regard as quite reasonable security policies.

Bob Morris: Mark, should I conclude that the Needham-Schroeder protocol works or should I conclude that it does not work?

Reply: I haven't given you sufficient information to determine that. What I'm really trying to say is because people copied them, subsequent protocols don't work, but it's not the fault of Needham-Schroeder.

Dieter Gollman: It's not the fault of Kerberos that it is used in the financial sector, because if you read the technical report there are big warning messages in front: this is the sign of this academic environment, don't use it in a highly sensitive environment like the financial sector.

Bruce Christianson: It doesn't work in the academic environment either, but that's a separate question.

Reply: Both the Needham-Schroeder protocol and Kerberos, and many others like it, take the view that you just don't create audit trails. There's an opposite extreme, which I will paraphrase as "if it moves, audit it". Anything that happens, if you make a record of it, it might come in useful later. This might

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 15–20, 2000.

sound extreme but some organisations have policies like that. Some of them have policies forced upon them, it's not that they make a conscious decision. If they are regulated as we are, we may get instructed by an external body that we must audit everything that we possibly could audit. So you have to think very carefully about the design of your system if you're going to be subject to that sort of constraint. I actually happen to think that there's a sensible compromise, at least in a financial institution, which is it's impractical to audit everything, but there's an awful lot that you should audit.

This may seem a bit of an aside but it will explain why multiple conflicting security policies may be forced upon you. This is a very, very condensed history of the company that I work for, Goldman Sachs. We started back in 1869 in New York and essentially the organisation remained within New York for a century, we didn't venture outside the United States. Any foreign, to the US that is, transactions were engaged in by negotiating with some foreign financial institution, which would then carry out the business on our behalf. Just over a century later they decided that they would go international and the office where I now work opened in 1970 in London.

I think it showed what their intent was that they didn't call it Goldman Sachs London, they called it Goldman Sachs International which is in fact the organisation that I work for. I think the opening of that office actually lulled the firm into a false sense of security because the regulations in London and New York are broadly similar. There are different organisations that will regulate us in both those locations, but the sort of requirements that we're given are roughly similar, so it was a reasonably painless exercise opening the London office. So we decided to expand further, we went into Tokyo and Zurich next.

Now Tokyo, although it seems to be a long way away, wasn't actually that much different, the regulations there were broadly similar to the London and New York office. Zurich is the first office which presented an interesting problem. Swiss banking regulations are such, that if you abide by the security policy that is thrust upon you by the Swiss regulators, you cannot comply with the American law or British, or the Japanese regulations. Similarly, this is a sort of reciprocal arrangement, you cannot enforce the New York, London and Tokyo regulations in Zurich. So we learned from that that there may actually be a little more to moving into other countries than we originally thought.

Bob Morris: From the point of view of Goldman Sachs, is there a country named Switzerland?

Reply: Well, there is a country called Switzerland, but Switzerland is divided into Cantons that have different banking regulations in each, which is why I refer to Zurich rather than Switzerland. They're trying to harmonise and to some extent they've harmonised.

This was deemed a success, so we now have over 90 offices. You'll notice the contraction here, it took us a hundred years to open the second office, we're now up to over 90. Surprisingly there weren't too many conflicts of the type I've described, but there were some. What I would observe is, policies are what I would call cultural, different countries have or institute policies that reflect

the culture of the organisation. One of our most recent offices is in Moscow, where it used to be said that everything is forbidden. Frankfurt said everything is forbidden unless it's to explicitly permitted. London is easy, everything is permitted unless it's explicitly forbidden. We also have an office in Milan where everything is permitted even if it's explicitly forbidden ...

The interesting offices from my point of view, at least from the point of view of this talk, are really Frankfurt and Zurich. They're the two that actually present me with the most problems. They're tiny by comparison to London or New York but they cause far more problems.

This is an oversimplification of policy which essentially applies in London and New York and also in Tokyo, which is the financial institution, assuming it's licensed, will hire its own firm of external auditors to audit the books, they will then give a report to the financial regulator and if the financial regulator is happy, he allows them to do business. If there are irregularities then the regulator may investigate. The interesting feature here is we are permitted to breach our confidentiality if we regard something as being suspicious. We are supposed to report it. If the regulators for some other reason have determined that something is suspicious, and they can justify that, they can ask us to breach our confidentiality, although we don't do that as a matter of routine. We would like to guard the privacy of our clients where it is reasonable and legal. It is a good idea, and since I'm responsible for this I actually insist upon it, that you have audit trails in these jurisdictions. It's not actually mandatory, but if you don't have them the regulator is entitled to ask why.

Here's why Frankfurt is potentially interesting, the regulator hires the auditors. In practice it seems to be the same audit firms, it's the big audit companies, except that they have a duty to the regulator and not to us. Having said that, the auditor is entitled to send us the bill for the auditing and we're not allowed to question that. We can only question that as a fact, we can't question their opinion as to what they think it is appropriate to spend.

Regulators may investigate anything they see fit, in particular they are entitled as a matter of law to breach the confidentiality of our clients. They can walk in to any of the offices that are subject to this jurisdiction and they may ask to look at any client records and we have a statutory duty to respond within 24 hours. Audit trails are an explicit regulatory requirement, I don't particularly mind the last one, but I mind the last but one of these rules, it worries me somewhat.

Zurich is interesting because it seems to go to the opposite extreme, which is interesting since these are both German speaking areas. Like London and New York, we are responsible for hiring the auditors, not the regulator. The regulators similarly can investigate irregularities, but they can't just walk in and demand to look at anything. In particular, they are not allowed to breach client confidentiality even if they have reason to believe that there is an irregularity.

Bob Morris: The question is whether what's going on violates foreign law in the country where the event originated.

Reply: I'm talking about something that takes place entirely within Zurich, just in that one City.

Bob Morris: Then I have no question.

Matt Blaze: The regulatory process here, is this for the purpose of investigating breaches of policy by you, or breaches of policy by your clients?

Reply: Yes. It's really the former, because the purpose of the regulator is to work out whether we're fit and proper to be a financial institution. This is whether we conduct our business appropriately. The point is that German and Swiss regulators take a different view as to what is legitimate to look at in order to satisfy yourself.

Frank Stajano: So presumably there will also be another set of regulations which you're not perhaps interested in talking about now which says what investigators can investigate in the case they suspect that your employer is doing something bad, which also depends on what the country is.

Reply: If it is entirely within Zurich then the rule is quite clear, we are not permitted to co-operate with the investigation of one of our clients. We might question whether they're appropriate as clients, but we're not permitted to reveal information to someone who's investigating one of our clients.

Stewart Lee: Is it not the case that there are certain exceptions to that?

Reply: That's the exception, if there was reason to believe that it's drug-related money.

Again I would advise the office to keep audit trails. However, they should bear in mind that client confidentiality takes precedence, which can be awkward. If you are, for instance, as we do, processing treasury transactions on a mainframe sitting in New York, because we are an international institution we try to have consistency, it would be prudent for us to audit everything, but there is a danger that we might fall foul of these regulations and we have to be very careful what is audited.

I would suggest that this supports the view that Stewart put forward earlier where he says that you can get incompatible policies. There's nothing you can do about it. The most extreme incompatibility is between Frankfurt and Zurich where there is an explicit right for a Frankfurt regulator to look at client information without any cause. In Zurich there is an explicit prohibition on the regulator looking at client information unless they have very, very good cause.

Further to this, because this is an example of something that Stewart said as well, Frankfurt policy is essentially more strict than New York and London policy. In principle you can force London and New York to conform to Frankfurt policy because that will satisfy the New York and London regulators, and generally I would assume, Japanese regulators.

However, what we actually do in practice. Enforcing the Frankfurt regulations would actually lead to an awful lot of complication, just for the sake of what, for my money, is a rather small office. The actual policy that we're subject to, that the regulators subject us to in Frankfurt, is more complicated than I stated earlier. It is primarily concerned with the actual receipt books and records. They're concerned with auditing in the old financial sense rather than

the information security sense, they want to know that books and records of our Frankfurt office are fair and faithful representations with relative certainty. Strictly to comply with the regulations, we only need to apply auditing in the information security sense to users who have access to journal books and records. However, and this is really what I'm going to conclude, we've determined that it is cheaper in practice to design a new security policy which is actually stricter than any of the policies that any of the regulators impose upon us. For instance, part of the Frankfurt policy says that financial data, client related financial data, going outside German Federal borders shall be encrypted. We could in theory identify that data and encrypt it, but that's a pain in the neck, what we in practice do is we say it's much easier to just encrypt the whole lot.

We are required to record certain audit events on certain machines, which again is concerned with books and records. It is much easier from the point of view of our system administrators if I say, please will you install these auditing events on all of the systems under your control because I'd rather not be lumbered with the task of keeping track of which systems are definitely concerned with books and records which are only indirect and subject to less strict regulation. This also protects us against a possible accident, that if we failed to identify a machine and therefore decided not to institute the appropriate auditing, we could again fall foul of the regulations. It is actually safer from my point of view if I do not make the attempt to identify those and just say, I'm going to be stricter than everything that I am required. So I would actually say that in a sense that's a slight counter-example to Stewart's earlier comments, it's not really, but it's saying that sometimes it may be appropriate to find a stricter policy and say, if everything else is a subset of that, then I solve the problem.

Stewart Lee: But you may have replaced the inconsistency with a key management problem because you're using a lot of encryption.

Bob Morris: Is it practical or even legal to send an encrypted record from Germany to France and back?

Reply: Yes, it is legal.

Bob Morris: That's new, because that's no older than about four years.

Reply: In fact it would be an offence not to.

Ross Anderson: But you also have the problem that when the European Data Protection Directive court cases finally come in four or five years time, you may well suddenly find that it's completely illegal to send German client financial data to New York under any circumstances.

Stewart Lee: There're encrypted aren't they, when you send some of the client data from Germany to New York? The details about who owns these millions of marks are encrypted.

Ross Anderson: Yes, but that's not sufficient to satisfy the European legislation.

Michael Roe: The issue is not about interception in transit, but in what your counterpart in the New York office, who is not subject to European law, then does with it.

Matt Blaze: I'm a bit confused. You convinced me a couple of slides back policies were inconsistent, and yet we don't see bankers being hauled off to prison. International banking seems to happen, and in fact you opened more offices rather than closing them down. You decided that it was possible.

Reply: The reason I'm not being hauled off is I'm trying to comply with the regulations.

Matt Blaze: But you just mentioned it's impossible to comply with the regulations. So that suggests to me that perhaps the regulations aren't in fact inconsistent, but there's some subtlety simply not modelled in the analysis.

Reply: It's not possible to apply all of the regulations consistently everywhere. I can tell you that because of the privacy rules in Zurich, we essentially treat transactions in Zurich as a special case, and I do not know the names of clients in our Zurich office. Our Treasury Department in London, which in theory is responsible for those transactions, does not know the names of clients. They will clearly ask for the appropriate audit report for Zurich saying that they believe that the Zurich office is applying for appropriate controls, otherwise they would be somewhat foolish.

Matt Blaze: So the policies aren't inconsistent, they don't meet, there is no policy that forces me to comply everywhere.

Reply: It would be much more convenient if I were not subject to all of those regulations, it would be useful to be able to apply a consistent policy across an organisation, we would like to be able to centralise some of the functions in our organisation, and we are forced not to because of the inconsistency.

Bob Morris: Won't the formation of the EU solve some or much of that?

Stewart Lee: That would be good, Switzerland isn't in the EU.

Bob Morris: Switzerland will always be outside, but many of the other countries you spoke of will be inside. Will that reduce the number of countries and the number of regulations you are subject to?

Reply: It already has, because the old Frankfurt regulations said that since no other country instituted rules as strict as the Germans, then clearly it would be unreasonable of you to send any data outside Germany. But they relaxed it by saying it could be sent out provided it's encrypted.

Virgil Gligor: What I think that Matt is pointing out is that if you have inconsistent policies and you are not forced to compose them, you may not necessarily have a problem, and what you are saying is that I like to compose them.

Reply: Yes.

Matt Blaze: Right now there's a consistent policy that says, if in Germany do this, if in France do this.

Jikzi: A New Framework for Secure Publishing

Ross Anderson and Jong-Hyeon Lee

Computer Laboratory, University of Cambridge
Pembroke Street, Cambridge CB2 3QG
{rja14,jhl21}@cl.cam.ac.uk

Abstract. We present a new way to deal with security policy on the web. Our aim is to let people build integrated publishing and e-commerce services using appropriate, simple and uniform mechanisms. Our goal is a single framework that applies equally to the publication of catalogues, music, software, public key certificates and even old fashioned books. Our prototype framework, Jikzi, supports multiple security policies – even in the same document – with a single transparent markup language.
Historically, most computer security research was motivated by military concerns; most papers deal with confidentiality, some with authentication. But the emphasis in commerce is on integrity, availability and accountability; confidentiality is often not a major concern at all. This motivates us to take a fresh look at the foundations of our discipline, and revisit not just security policy models but also the authentication and integrity mechanisms we use.
The growing importance of XML may simplify much; if it is adopted as widely as some expect, then everything from digital certificate formats through legacy multilevel secure systems will also be up for review. We believe our prototype system indicates a fruitful alternative approach.

1 Introduction

The security of electronic commerce can be thought of as three separate but loosely related problems – publishing, payment and copy control.

Firstly, most of the activity which people describe as electronic commerce consists of some kind of publishing. Publishing used to be understood as a business sector – the business of making and selling books, CDs, videos and software – but it increasingly makes up the very stuff of online business. Regardless of the product or service sold, electronic commerce consists largely of publishing web pages, catalogues and price lists, as well as more specialist items such as public key certificates.

Secondly, once an online customer has decided to purchase a product or service, the deal is typically done using a credit card. As the cardholder is not physically present, there are a number of risks (such as the use of stolen credit card numbers) that are traditionally controlled by such rules as 'goods bought by mail or telephone order must be dispatched to a cardholder address verified with the credit card issuer'. However, transactions may now involve the delivery of intangible goods to an email address, and this may create new risks.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 21–36, 2000.

Thirdly, it is possible for customers to make perfect copies of digital goods such as music and software, and pass these on to their friends. Pirates may even mass manufacture them and sell them at a discount. How can copying be controlled, given the open nature of the medium and the difficulty of making consumer electronic devices really tamper-resistant?

The second and third problems have been studied extensively. The industry appears to have settled on SSL/TLS as a standard for online credit card payments; meanwhile, there are many moves to control copying, ranging from the launch of new digital audio formats to an upgraded version of DVD. However, the first problem – secure publication – has received little attention.

Yet secure publication is important. Although possible embarrassment from a hacked web site might be the first risk that springs to mind, there are much worse financial exposures (e.g., if a stock manipulator can pass off bogus news about a non-existent merger or acquisition). There are also serious safety hazards: for example, doctors are starting to rely on online reference sources for data on drug doses and reactions.

In this paper we describe a security policy model that seeks to deal in a quite general way with the issue of secure publication, whether of books, music, software, public key certificates, videos, drug formularies, aircraft maintenance manuals, academic papers or business records. We describe a set of mechanisms to support authentication and integrity in a quite general way in such environments, with a particular emphasis on audit and accountability. We sketch the beginnings of an authentication logic to help us analyse whether a particular protocol is sound. Finally, we describe a prototype implementation, which we call Jikzi, after the first ever book published using a movable type printing press; this is a Buddhist text printed in Korea in 1377, some 63 years before Gutenberg.

2 Previous Work

The direct precursor of Jikzi was the Wax system, developed to secure the online publication of medical information such as drug data [4]. The main problems we encountered were the performance penalty imposed by trying to verify the signature on a whole book when the reader simply wished to consult a single section of it and the difficulty of securing long-lived data such as books using relatively short-lived objects such as public keys. Wax solved these problems by associating with each publisher a tree of hashes: the leaves were hashes of the individual sections, and the nodes progressed up through chapters and books until the root, which protects a publisher's whole catalogue, is authenticated by other means (typically a one-time signature). Thus when a user opens a book at the entry corresponding to a particular drug, the entry can be verified quickly without having to check a signature on the entire book. The lesson learned from Wax was that it is a sound design principle in publishing to use hash trees and to minimise the amount of signature – preferably to a single signature on each new version of a publisher's catalogue.

The Wax design has had some success in the UK medical informatics field. But it uses a proprietary hypertext format, so the next step was to generalise it to work with html. The result was the ERL, or eternal resource locator [3], which can be thought of as a URL plus a hash of what you expect to find there. Having developed this, we realised that it was very powerful and useful: the writer of a web page or other document can authenticate any digital object by simply including its ERL, and this enables the construction of webs of trust of the kind familiar from PGP but applying quite generally to objects rather than just to the names of principals. Many of the problems associated with public key infrastructures simply go away; one ends up managing a few root keys and doing a lot of version control.

It has now become clear that we need to support XML as this is the emerging framework for commercial markup languages [8]. XML is well positioned between SGML and HTML; it is easier to use than SGML, yet and provides more degrees of freedom in extension than HTML. Major software vendors are committed to supporting it, including both major players in the browser market, who already support XML processing partially and are expected to support it fully in the near future.

The goal of the Jikzi project is thus to develop general XML-based mechanisms for embedding security properties in hypertext documents, and to understand the underlying computer science issues such as resilience, scalability and verification.

This brings us to other work in the same general field. Two particular projects bear mention: the Secure Document Markup Language (SDML) developed by Kravitz at IBM [17] which defines the tags necessary to implement electronic cheques ('payee', 'amount', 'currency', 'signature', 'certificate') and the World Wide Web Consortium's DSig project [10] to implement digital signature technology in content marking and rating schemes. While both of these projects are useful and instructive, we need something more general; we want to deal not just with cheques but with all other kinds of bills of exchange (such as bills of lading) and we want to be able to deal with inheritance properties (a bank cheque is a kind of cheque, and a cheque is a kind of bill of exchange).

A general security system should also be able to deal with applications such as military messaging: governments are likely to remain wedded to the idea of having paragraph-level security labels (i.e., a document is mixture of paragraphs at different levels such as unclassified and secret), and XML appears to be the only serious way in which the relevant document management routines can be implemented without rewriting a huge amount of software.

We also ought to be able to deal with publishing in the most general sense, which means dealing with issues of replication, persistence and control. Previous work in this area includes the Distributed Object-based Document Architecture (DODA) [25] which is aimed at computer supported cooperative work, such as software development. DODA is somewhat like a secure RCS in which 'folios' of documents are bundled together and managed by hash and signature mechanisms according to a policy such as 'a software upgrade can only be issued when

both a testing manager and a customer manager sign it'; such policies can be enforced using mechanisms such as threshold signature.

3 A New Conceptual Model

Academic security research has tended to be 90% on confidentiality, 9% on authentication/integrity and perhaps 1% on availability. Yet commercial information systems departments have the opposite priorities. Availability of service is tops, with perhaps a third of the whole budget going on hot spare backup sites, network redundancy, and so on; integrity is next with several percent of the budget going on consistency checking mechanisms, internal audit and the like; but confidentiality is not a high priority at all [1]. In the world of publishing, this is especially so. So what would a security policy model for publishing look like?

At its simplest, a publisher is a principal who from time to time creates and makes world-readable a document such as a book, and also from time to time creates and makes world-readable a catalogue or index of such documents.

Stripping this to the bare essentials, we posit a universe in which each user can append data to a single file, which all other users can read. Deletion of data – even in one's own file – is not allowed. Thus we exclude consideration of confidentiality for the time being; the integrity of all strings once written is guaranteed; the authenticity of the author of a string is obvious; and we make digital signatures trivial (I can sign anything by simply asserting it in my file). The immutability of data once written means that a user can countersign a document by reference, e.g. 'I hereby concur with document number 382 in Bob's file'. We will ignore complex data structures such as indexes, as the mechanisms whereby they can be created from flat append-only files are well known.

We will now discuss append-only file systems and their implications for modeling security policy.

3.1 Append-Only File Systems

Append-only file systems are rarely considered in the research literature, but are remarkably common in real life. Banks must keep records for a period of time set by law (6 years in the UK), and much the same applies to medical records, though the period for which a record must be kept depends on the condition to which it relates – cancer records, for example, are kept for the patient's lifetime [2]. Business records in general are kept for the period during which transactions can be challenged by customers or audited by tax inspectors.

Publishing is perhaps the oldest application of append-only file systems. We do not expect that the archives of a newspaper or journal will ever have their content changed (except in the shrinking number of totalitarian states); if an issue of a publication is found to be defamatory or in breach of copyright, the appropriate legal remedy is financial compensation rather than retrospective modification.

Similarly, in commerce, the need for finality of transactions dictates that if a bank transaction is processed in error, it is not retrospectively deleted but rather a correcting transaction is passed through the system. (In the days when bank ledgers were kept on paper, they were kept in ink using bound books.)

There are many mechanisms which can be used to implement an append-only file store, ranging from CD-WORM drives through RCS to mainframe system managed storage products. In some applications, such as retail banking, the volume of transactions is large and thus the disk capacity required is fairly predictable; in others, such as medical records, the quantity of data which a doctor can type into his PC is negligible compared with the size of even the smallest hard disk; in others, there might be some risk of denial-of-service attacks involving resource exhaustion, and so one might either use disk quotas or insist on micropayments for disk writes.

The academic literature on append-only file systems, however, appears to be rather sparse. The earliest mention of which we are aware is the use of a public bulletin board by Benaloh to ensure the serialisation of pseudonymous votes cast in a digital election [6]. More generally, log structured file systems improve the performance of some systems by ensuring faster writes at the expense of longer read times [23,24]. In systems in which erasure is expensive, such as flash memory, they allow an even distribution of writes in the media [16]. We might also mention the Eternity Service [1], a distributed file store in which techniques of fragmentation, redundancy, scattering and anonymity are used to provide the highest possible level of protection against service denial attacks: in effect, an append-only file store which is highly resistant to deletion attempts.

3.2 Example Application

The reason we choose to use an append-only file system as our fundamental primitive is that it greatly simplifies many applications involving availability, integrity and authentication properties. By so doing, it forces us to think clearly about what optimisations we are making, and why, when we depart from it.

A good example is the B-money system proposed by Wei Dai [12]. It assumes that each user maintains a separate account of how much money everyone in the system owns, and in order to make a payment, one simply broadcasts a signed message saying 'I, Alice, hereby pay the sum of X to Bob'. B-money assumes a synchronous, unjammable broadcast channel, but can clearly be implemented in our model by having the principals append the transaction to their file. Secure serialisation is required but can be implemented in the obvious way using Lamport time [18] (but see below).

B-money has some interesting properties. For example, if someone computes the account balances of other parties incorrectly, it is himself that he puts at risk; he might accept a payment where the payer had insufficient funds, and then be embarrassed when one of his own payments was rejected by a more careful principal on the grounds that he in turn could not cover the payment.

This is a surprisingly apt model of a payment system: in the banking world, journal files are used to accumulate transactions from cash machines, cheque

sorters, teller stations etc and these are applied overnight to the account master file in order to update it. The journals fill the same role as the transactions in b-money, and the overnight run is an optimisation which simplifies processing. To the extent that a bank's functions are mechanical (rather than to do with credit assessment and risk management), it might be replaced by a b-money system plus a software agent that performs this overnight update.

3.3 Significance

A security policy model, such as Bell-LaPadula [5] or Clark-Wilson [11], is a simplified way of looking at system security properties. It usually corresponds only partially to the reality of fielded systems, but is instructive in that the optimisations in these systems become clear for what they are.

The Jikzi model of the universe, as a set of append-only files, fulfills exactly this function. If the persistence and integrity of other principals' records could be relied on, then the design of business record systems would be much simpler; as it is, a prudent business will not trust all its counterparties to keep full records and make them available timely.

Thus instead of trusting that Bob's file is truly append-only and countersigning by reference ('I hereby concur with document number 382 in Bob's file'), a businessman in the real world will countersign by value ('I hereby concur with Bob's statement X and cite his signature S') or hash ('I hereby concur with Bob's statement whose SHA-1 is X and cite his signature S on it'). Whether one retains a copy of the message, or just the hash, will depend on whether one expects to have to produce the message at some time in the future, or whether the production would only be of value to another party (in which case there are trust models in which one still wants to retain the hash in order to have a complete list of the signatures one has ever made with a given key [22]).

So our model is not just a (relatively) trivial way of modeling what happens in an ideal electronic publishing house, but also a principled way of looking at business records. Previously, the main security policy model proposed for commercial computer systems – Clark-Wilson – focussed entirely on the transactional aspects (e.g., how to ensure that it took two employees working together to raise a cheque, and that an indelible log of their actions was kept). The external record keeping aspects have so far escaped, perhaps because until very recently all business records were kept in paper form (even if the working copies were electronic) and so the issue of managing evidence in electronic form was just not considered. However, as electronic commerce spreads, this issue will become ever more important, and the Jikzi model provides a way of understanding it – that I must keep a copy of that part of the universe on which I rely[1].

Finally, work on the legal validity of electronic signatures has so far focussed on the signature itself, type approval of the signature creation device, licensing

[1] There are some aspects of the universe that it might not be practical to keep; if I have an old VisiCalc file, do I need to keep an Apple II computer in working order so that I can browse it? However, our model can at least help people identify and assess such difficult dependencies.

and insurance of CAs, and so on. However, this is only the tip of the iceberg; context, and in particular the relationship between documents, is vital in real court cases. Did the hospital's web site list the side effects of drug X when the plaintiff visited it? Did it point to version 1.27 or 1.28 of MegaDrugCorp's fact sheet at the time? In general, who knew what when, which document was signed first, and how does the chain of evidence hang together?

It is for reasons like this that a prudent organisation, seeking to archive its business records, will not just timestamp and store html pages; we need some means of protecting – and auditing the use of – links and other dynamic aspects of the content. Here, the Jikzi approach comes into its own.

4 Towards an Authentication Logic

As this is a protocols workshop, it is natural for the reader to ask what sort of authentication protocols are natural in the Jikzi model. In fact, if the append-only model takes care of all integrity and authenticity concerns, do we need any protocols at all?

The answer, perhaps surprisingly, is yes. Time is not included in the basic security model, and so the fact of one event's having happened after another must be established explicitly. This is also true in conventional publishing, where dependence is typically established by citation. But even this is not straightforward; there are many cases of papers in a learned journal citing each other, and in general, where something predictable is going to be written, it can be cited in advance of its publication.

Thus in order to be sure that someone has read a message which I have written, I should include in it something unpredictable, such as a random number. In this way, incontrovertible serialisation can in principle be established, and constructs such as Lamport time can follow.

Suppose that Alice wants to ensure that after she makes some statement X, say, Bob reads it before responding Y. Our first attempt at such a protocol was:

$$A \longrightarrow B : B, X, N_B$$
$$B \longrightarrow A : A, Y, h(X, N_B)$$

Here, the element '$A \longrightarrow B : B, X, N_B$' means that A writes, for B to read, the values B, X and N_B in succession.

We looked at a number of verification logics to see which of them might be adapted to our purpose. The primitive we need seems to be something like:

B **said** Y **having seen** X

and this should be derived from a rule somewhat like the following:

B **said** (X, Y), X **unpredictable** *implies* B **said** Y **having seen** X

The elaboration of this into a formal verification system is work in progress; we report it in the tradition of the Cambridge protocols workshop. There are two interesting things that we have noticed so far. Firstly, in Kailar's logic of

accountability [15], we cannot distinguish between different instances of a receipt; thus the verification of the IBS protocol [20] is incomplete, as IBS is silent on whether receipts have serial numbers (in fairness, its successors did have; the point is that the verification might usefully have dealt with them). Secondly, in the BAN logic [9], a nonce is a nonce: any distinction between a random challenge, a serial number and a timestamp is made in the protocol idealisation stage. Yet here, uniqueness and unpredictability are definitely different. What is more, we might want to have a rule saying something like:

B **said** Y **having seen** X, X **unpredictable** *implies* Y **came after** X

But it is not quite that simple. For example, if Alice quotes a message of mine which I know to have been unpredictable because it contained a random number which I generated, then I know that she read it; however, Bob might suspect me of having colluded and told her the random number. So unpredictability is in the eye of the beholder, and we may have to refine our rule to something like:

B **said** Y **having seen** X, A **believes** X **unpredictable** *implies*
A **believes** Y **came after** X

We would not claim that a short discussion of the rules a suitable logic might contain, amounts to the definition of a logic (that is for future work). However, even the discussion thus far leads us directly to our first 'scalp': the above draft protocol is wrong. The second message should be

$$B \longrightarrow A : A, h(X, N_B), Y$$

This was not at all obvious at the outset, and it is not clear how the mistake could have been found with an existing tool.

A final point, which we discussed at the workshop presentation but do not have the space to elaborate here, is that when one combines the Jikzi policy model with software agents (such as the agent we mentioned as a possible implementation of a bank in the b-money scenario above) then the result is a model that is somewhat similar to Clark-Wilson but somewhat more precise. We plan to develop this theme in a later paper.

5 The Jikzi Prototype

As mentioned above, one purpose of the Jikzi project was to extend the previous prototypes of securing documents in HTML and SGML such as ERL and SDML, using XML. XML enables the automation of document handling and interpretation using DTD (document type definitions), CSS (cascading style sheets) [7] and XSL (extensible stylesheet language) [13].

Jikzi uses its own markup language based on XML to support basic security features such as hashing and signature. The fastest way to explain it is probably by example, so we now describe how to write a electronic checque. First, we define a set of entities which will be used throughout our DTD files: `stdDef.dtd` contains the basic entity definitions. It also includes declarations for DTD files.

```
<!-- stdDef.dtd: DTD for entities -->
<!ENTITY % nameList "foreName CDATA #REQUIRED
    surName CDATA #REQUIRED
    initial CDATA #IMPLIED">
<!ENTITY % dateList "year CDATA #REQUIRED
    month CDATA #REQUIRED
    day CDATA #REQUIRED
    hour CDATA #IMPLIED
    minute CDATA #IMPLIED
    second CDATA #IMPLIED">
<!ENTITY % algoList "algoName CDATA #REQUIRED
    version CDATA #IMPLIED">
<!ENTITY % orgList "orgName CDATA #REQUIRED
    dept CDATA #IMPLIED
    section CDATA #IMPLIED">
<!ELEMENT dtd (dtdInfo)+>
<!ELEMENT dtdInfo EMPTY>
<!ATTLIST dtdInfo name CDATA #REQUIRED
    version CDATA #REQUIRED
    URL CDATA #IMPLIED
    author CDATA #IMPLIED>
<!-- end of stdDef.dtd -->
```

`signList.dtd` is used for signing a document; it contains signature information, plus the public key information needed to check the signature.

```
<!-- signList.dtd: DTD for signing docs -->
<!ENTITY % stdDef SYSTEM "http://www/~jhl21/dtds/stdDef.dtd">
%stdDef;
<!ELEMENT signList (sign)+>
<!ELEMENT sign (signInfo, pKeyInfo, signature)>
<!ELEMENT signInfo (signer, signAlgo, url?, parent?)>
<!ELEMENT signer EMPTY>
<!ATTLIST signer %nameList; orgName CDATA #IMPLIED
    signerId CDATA #IMPLIED>
<!ELEMENT signAlgo EMPTY>
<!ATTLIST signAlgo %algoList;>
<!ELEMENT url (#PCDATA)>
<!ELEMENT parent (#PCDATA)>
<!ELEMENT pKeyInfo (pKeyVersion, cert, pKey)>
<!ELEMENT pKeyVersion (#PCDATA)>
<!ELEMENT cert EMPTY>
<!ATTLIST certIssuer CDATA #REQUIRED
    certSerial CDATA #REQUIRED
    certUrl CDATA #REQUIRED
    revokeUrl CDATA #REQUIRED>
<!ELEMENT pKey (#PCDATA)>
<!ELEMENT signature (#PCDATA)>
<!-- end of signList.dtd -->
```

`eCheque.dtd` is a simple electronic cheque definition, and uses `signList.dtd`. It includes the cheque number, account number, payee, payer(s), payment amount, issue date, and timestamp. (A currency code can be specified with the payment amount.)

```
<!-- eCheque.dtd: DTD for electronic cheques -->
<!ENTITY % signList SYSTEM "http://www/~jhl21/dtds/signList.dtd">
%signList;
<!ELEMENT eCheque (dtd, chequeBody, signList)>
<!ELEMENT chequeBody (chequeId, account,
    payee, payment, issueDate, notLater?, timestamp, signInfo+)>
<!ELEMENT chequeId (#PCDATA)>
<!ELEMENT account (#PCDATA)>
<!ELEMENT payer (#PCDATA)>
<!ELEMENT payee EMPTY>
<!ATTLIST payee %nameList; payeeId CDATA #IMPLIED>
<!ELEMENT payment EMPTY>
<!ATTLIST payment amount CDATA #REQUIRED
    currency CDATA #IMPLIED>
<!ELEMENT issueDate EMPTY>
<!ATTLIST issueDate %dateList;>
<!ELEMENT notLater EMPTY>
<!ATTLIST notLater %dateList;>
<!ELEMENT timestamp (#PCDATA)>
<!-- end of eCheque.dtd -->
```

Electronic cheques are thus, as in SDML, just like paper cheques. There are external dependencies, or course, such as the certificates (which need to link the account number to the signer's distinguished name). In a real implementation, the dependencies are likely to get so complex that formal tools may be advisable in order to check for properties such as completeness and consistency.

Anyway, an actual electronic cheque looks like this:

```
<!-- corpCheque.xml -->
<?xml version="1.0"?>
<!DOCTYPE eCheque SYSTEM "eCheque.dtd">

<eCheque>
<dtd>
   <dtdInfo name="stdDef.dtd" version="1.0"/>
   <dtdInfo name="signList.dtd" version="1.0"/>
   <dtdInfo name="eCheque.dtd" version="1.0"/>
</dtd>
<chequeBody>
    <chequeId>00883627</chequeId>
    <account>23-45-67 1234567</account>
    <payer>University of Cambridge</payer>
    <payee foreName="William" surName="Hopkinson" initial="F"/>
    <payment amount="19.95" currency="UKP"/>
```

```
    <issueDate year="1999" month="01" day="15"/>
    <notLater year="1999" month="06" day="30"/>
    <timestamp>872043082393</timestamp>
    <signInfo>
        <signer foreName="John" surName="Smith" initial="M"/>
        <signAlgo algoName="PGP-RSA" version="5.5"/>
    </signInfo>
    <signInfo>
        <signer foreName="Edward" surName="Thompson" initial="J"/>
        <signAlgo algoName="PGP-DSS" version="5.5"/>
    </signInfo>
</chequeBody>
<signList>
<sign>
<signInfo>
    <signer foreName="John" surName="Smith" initial="M"/>
    <signAlgo algoName="PGP-RSA" version="5.5"/>
</signInfo>
<pKeyInfo>
    <pKeyVersion>1.0</pKeyVersion>
    <cert certIssuer="Cambridge Certificate Agency"
        certSerial="1234567890"
        certUrl="http://www.cca.com/certs/1234567890"
        revokeUrl="http://www.cca.com/revoke?sn=1234567890">
    <pKey>
    lQMFEDWhboWuyrPDhRvRXQEBkp4D/ivwpsci5MJQXUA
    bcPOUQquOgzMpp7W5KXP1Cit9EyqaPtet+1nkaoRXYv
    FQIB/eBjkcvNaA02w/mvHQRQYiAzz6kdPSn/rt9THkX
    LAOsOekv
    =1zy8
    <pKey>
</pKeyInfo>
<signature>
CZ/SDEjG6wt7V3uXWbZGV0pVg5LJg8j7bONjtdDuAHy
asD8dsMrWe82J23Kwe7sd2jh2348fsKS92R82kw/Tus
IyeYFI87qHE=
=OTeM
</signature>
</sign>
<sign>
<signInfo>
    <signer foreName="Edward" surName="Thompson" initial="J"/>
    <signAlgo algoName="PGP-DSS" version="5.5"/>
</signInfo>
<pKeyInfo>
    <pKeyVersion>1.0</pKeyVersion>
    <cert certIssuer="Cambridge Certificate Agency"
        certSerial="2345678901"
        certUrl="http://www.cca.com/certs/2345678901"
        revokeUrl="http://www.cca.com/revoke?sn=2345678901">
```

```
    <pKey>
    SHllb24gTGVlICgxMDI0KSA8Sm9uZy1IeWVvbi5MZWV
    trSrLDLzXysRlsCHis29Q74wmeTysqY3j2z+RtzAgXb
    ErsHSe7p3Jk23Ks23RksE89wEn32Zy7gwl29rt3l9/S
    L12s7ejk
    IEz1y
    </pKey>
</pKeyInfo>
<signature>
E0a57bT2+xWWds0Jh3wpIqV25B6+ExJA6xnAB3Az5hd
XZC36FgshDjRks72EosTNmsd7Us4ePgsQ/ZeX82HHiQ
xAEALBQYiHd
n/rt9
</signature>
</sign>
</signList>
</eCheque>
<!-- end of corpCheque.xml -->
```

(In the above example, signatures are truncated to save space.)

6 Architecture

In the Jikzi prototype, all documents published can be protected from modification and deletion by making the underlying document storage append-only. However, if such mechanisms could be relied on, then – as in our abstract security policy model – there would be no need for digital signatures, hashes and other such modification defection mechanisms. What we see such mechanisms actually achieving is to build, on top of a less reliable set of platforms, a document store that is more reliable by virtue of the documents, and the links and other relationships between them, being protected using hashing and digital signature mechanisms.

Ideally, a user should be able to follow the dependencies automatically and identify all of the documents on which she relies to support her business records, or to provide a fully documented medical record. This might not just include (for example) a copy of her record at her GP's surgery, but also copies of hospital records to which a discharge summary in the GP record referred, and copies of the relevant entries from drug formularies or treatment protocols on which particular treatment decisions were based.

This is of course an ambitious program, and prototype currently just provides us with the means to play with security markup languages, DTDs and cascading style sheets, so that we can begin to explore this space.

The Jikzi server consists of the following blocks: Server User Interface, Jikzi PreProcessor, Version Manager, Storage Manager, Policy Manager, Search Engine, and storage. The Server User Interface has a web interface format so that users can access it with their browsers; pages for user document submission are

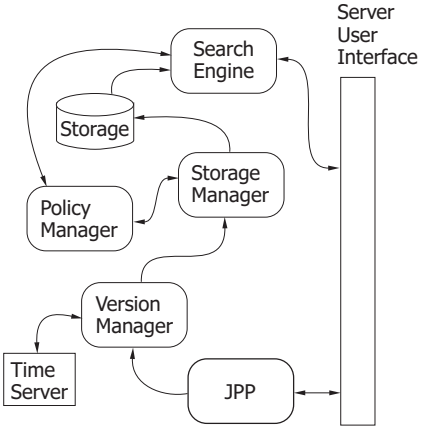

Fig. 1. Architecture of the Jikzi system

are written in HTML/XML; and the documentation for the Jikzi service is written in XML. At this stage, a limited number of browsers support XML features and we believe that support of XML will expand.

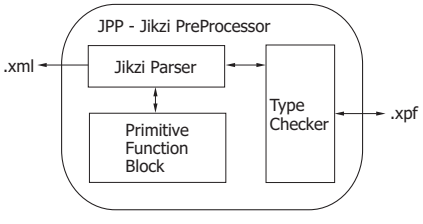

Fig. 2. Jikzi PreProcessor

The Jikzi PreProcessor (JPP) consists of three parts: the Type Checker, the Jikzi Parser, and the Primitive Function Block. The JPP's input is in XML format and it is tested, manipulated, and then stored in an internal format called XML Primitive Format (XPF). The Type Checker verifies that submitted documents are well-formed. The Jikzi Parser catches Jikzi specific tags and embeds the function they need into the document. The Primitive Function Block is a library of functions used in Jikzi server which includes hashing, signatured, and verification algorithms.

The Version Manager acts like RCS; it does version control of all published files and keeps traces of their update history. Files are stored in a log-structured way for efficiency. It also lets users browse the history of published documents.

The Storage Manager is a front-end to the file system and enforces the access control policy laid down by the Policy Manager. The Policy Manager is

a directory for policies and supports their registration and deregistration. It also interacts with the Search Engine. The Search Engine fetches documents requested by users on the basis of string matching. The Storage database, at least in our prototype, keeps all published documents in an append-only file.

The Time Server enables users to affix timestamps to publications. It can use either an external time service like the Network Time Protocol server [19] or an internal clock. At present, we use the latter.

This is still an alpha system and under development. Its purpose is to enable us to understand the issues involved with security markup languages, and the more input we can get, the better. The system is online at <http://han.al.cl.cam.ac.uk>, and if you want to be a tester, please contact the authors.

7 Conclusion

The three main aspects of electronic commerce that need to be protected against error and attack are publication, payment and copy control. So far the first of these has been largely ignored; in this paper we have tried to highlight the issues.

We have proposed a security policy model for publishing: a file store in which each user has append access to exactly one file, and all files are world readable. We have shown how some simple applications fit this model, and indicated its more general usefulness. We have discussed possible rules for an authentication logic that would enable us to reason about trust relationships. This analysis helped us to find an ambiguity in the previous work, namely how one deals with different instances of a receipt.

We have also described a prototype system that we have developed to explore the use of authentication and integrity mechanisms in XML. Jikzi is an electronic publishing system that provides long term storage with support for XML and CSS, and features such as version management and timestamping.

Our work has some other interesting scientific aspects. It may help to elucidate our understanding of time and succession in the context of protocols and authentication logics, and the relationship between availability and authentication in scenarios where some views of a database are preserved and others lost. This may give us a better understanding of application level trust structures. Until now, models such as Clark-Wilson, Rampart [21] and Proactive Security [14] have tended to consider 'dual control' in the context of simple threshold schemes; but in real applications, one tends to find a much richer syntax of dependencies that are integrated with the underlying business model: it may take the assent of a number of principals with distinct roles to commit to a transaction, and a different set of principals to override it or recover from failure. This can give much greater resilience, but at a cost of more complexity – complexity familiar to commercial application developers, but which the research community has so far largely ignored.

The arrival of XML is bound to force application security syntax to be examined, and Jikzi provides the context in which this can be done.

8 Acknowledgement

The second author is supported by the EPSRC under grant number GR/L95809 on Resilient Security Mechanisms.

References

1. Ross John Anderson. The Eternity service. In *Pragocrypt '96*, pages 242–252, Prague, 1996. CTU Publishing House. 24, 25
2. Ross John Anderson. Security in clinical information systems. BMA Report, British Medical Association, January 1996. ISBN 0-7279-1048-5. 24
3. Ross John Anderson, Vaclav Matyáš, and Fabien Albert Pierre Petitcolas. The Eternal Resource Locator: An alternative means of establishing trust on the world wide web. In *1998 USENIX Electronic Commerce Workshop*, pages 141–153, Boston, MA, 1998. 23
4. Ross John Anderson, Vaclav Matyáš, Fabien Albert Pierre Petitcolas, Iain E. Buchan, and Rudolf Hanka. Secure books: protecting the distribution of knowledge. In *Proceedings of Security Protocols Workshop '97*, pages 1–12, Paris, 1997. 22
5. D. Elliot Bell and Leonard J. LaPadula. Secure computer systems: Mathematical foundations. Mitre Report ESD-TR-73-278 (Vol. I–III), Mitre Corporation, Bedford, MA, April 1974. 26
6. Josh Cohen Benaloh. *Verifiable Secret-Ballot Elections*. Ph.D. dissertation, Yale University, New Haven, 1987. YALEU/DCS/TR-561. 25
7. Bert Bos, Hkon Wium Lie, Chris Lilley, and Ian Jacobs. Cascading Style Sheets, level 2 (CSS2). W3C Recommendation REC-CSS2-19980512, World Wide Web Consortium, May 1998. `http://www.w3.org/TR/REC-CCS2`. 28
8. Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML 1.0). W3C Recommendation REC-xml-1998210, World Wide Web Consortium, February 1998. `http://www.w3.org/TR/REC-xml`. 23
9. Michael Burrows, Martín Abadi, and Roger Michael Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990. 28
10. Yang-hua Chu, Phillip DesAutels, Brian LaMancchia, and Peter Lipp. PICS Signed Labels (DSig) 1.0 specification. W3C Recommendation REC-DSig-label-19980527, World Wide Web Consortium, May 1998. `http://www.w3.org/TR/REC-DSig-label`. 23
11. David D. Clark and David R. Wilson. A comparison of commercial and military computer security policies. In *The 1987 IEEE Symposium on Security and Privacy*, pages 184–194, Oakland, CA, 1987. 26
12. Wei Dai. B-money, 1998. `http://www.eskimo.com/ weidai/bmoney.txt`. 25
13. Stephen Deach. Extensible Stylesheet Language (XSL). W3C Working Draft WD-xsl-19990421, World Wide Web Consortium, April 1999. `http://www.w3.org/TR/WD-xsl`. 28
14. Amir Herzberg, Markus Jakobsson, Stanis law Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *The 4th ACM Conference on Computer and Communications Security*, pages 100–110. ACM SIGSAC, 1997. 34
15. Rajashekar Kailar. Accountability in electronic commerce protocols. *IEEE Transactions on Software Engineering*, SE-22(5):313–328, May 1996. 28

16. A. Kawaguchi, S. Nishioka, and H. Motoda. A ash-memory based file system. In *The 1995 USENIX Technical Conference*, pages 16–20, New Orleans, LA, January 1995. 25
17. Jeff Kravitz. SDML – signed document markup language. W3C Note NOTE-SDML-19980619, World Wide Web Consortium, June 1998. `http://www.w3.org/TR/NOTE-SDML`. 23
18. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978. 25
19. David L. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, COM-39(10):1482–1493, 1991. 34
20. K. R. O'Toole. The internet billing server transaction protocol alternatives. Technical Report INI-TR 1994-1, Carnegie Mellon University Information Networking Institute, April 1994. 28
21. Michael K. Reiter. The Rampart toolkit for building high-integrity services. In *Theory and Practice in Distributed Systems*, volume 938 of LNCS, pages 99–110. Springer-Verlag, 1995. 34
22. Michael R. Roe. *Cryptography and evidence*. Ph.D. dissertation, University of Cambridge, Cambridge, 1998. 26
23. Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. In *13th ACM Symposium on Operating Systems Principles*, pages 1–15, Asilomar, October 1991. 25
24. Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. *ACM Computing Surveys*, 10(1):26–52, 1992. 25
25. Jean Fiona Snook. *Towards Secure, Optimistic, Distributed Open Systems*. Ph.D. dissertation, University of Hertfordshire, Hatfield, September 1992. Technical Report No. 151. 23

Jikzi: A New Framework for Secure Publishing
(Transcript of Discussion)

Ross Anderson*

Computer Laboratory, University of Cambridge

Jikzi was the first ever book published using a movable-type printing press, it was manufactured in a temple in Korea in 1377 and thus predated Gutenberg by a generation.

Now this is work that I've done with Jong-Hyeon Lee who's one of my research students. What's it about? Well, we were told that we needed a talk that was relevant to the entertainment industry and also had a strong audit content. Jikzi is a system that is suitable for entertainment in that it is a system for publishing, and publishing arbitrary things, anything from the latest rock-n-roll single through to a great big wadge of public key certificates. What we're trying to do is build a general purpose publishing system, and there is a heavy emphasis on accountability, on being able, if you are a customer, to be pretty well certain that the instance of the book that you now have sitting in your browser is a valid instance of the book, that it is timely, that it has not been tampered with, and so on and so forth.

Now this brings us to an interesting use here of the policy model, which will pick up on some of the things that we were arguing with Stewart on Monday. There are authentication and integrity mechanisms of course, and one of the interesting things that we're just starting to develop is a new authentication logic which sits in the hierarchy of things somewhere between Kailar and the BAN logic, in that it does strictly more than Kailar and strictly less than BAN, and, unlike many other things, we've got a real implementation underway and it's fully buzzword compliant.

Where does it come from? Well, at the protocols workshop four years ago I introduced the eternity service: a distributed filestore, which uses anonymity, duplication and scattering, in order to scatter a file over the Internet in such a way that it becomes highly resistant to denial of service attacks. And that's spawned a number of implementations and a mailing list. There was traffic on the mailing list just this morning, people were discussing how you go about managing directories in eternity. Well, we've got some ideas on that.

What we were trying to do with eternity was to replicate the politically important feature of book publishing, namely, once you have published the Bible, or whatever, in the local language, and you have 50,000 copies of it out there, then the Archbishop of Canterbury can't just round them up and put them in the fire and put you in the fire with them, as happened with people who tried to publish the Bible before movable type printing came to the west. Incidentally, Jikzi was also a religious tract; it was a Buddhist religious book.

* Joint work with Jong-Hyeon Lee

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 37–47, 2000.

So, publishing has politically important features, namely that things can't be unpublished. It's also got integrity features, in that if you go to a random bookshop and you buy a copy of a book, the chance that this has been maliciously tampered with in some targeted way by your personal opponent is something that you can generally ignore. And we've done some work based on this, we've published a global trust register, a book containing all the important public keys in the world. That's now been taken over by MIT Press.

Another thing, that we did with Fabien [Petitcolas] and Vaclav [Matyáš], was to ask the question: how do you go about publishing a medical book on-line? The particular medical book that we've done most with is the British National Formulary, which is a book published every six months that lists all the drugs that can be prescribed in Britain together with the approved dosages, side effects, cross effects, and so on. Now if you were pulling this up into your browser at Addenbrookes as you were sitting there writing a prescription for somebody, you need to know that the information is genuine. And you're perhaps not worried so much about malice as about all the things that go bump in the night with computer systems, because in the context of an on-line book, these can kill people. Wax was a proprietary first attempt at doing this. We started off with a request from Rudolph Hanka, the Head of the Medical Informatics Unit: we've got this on-line publishing application, can you put some integrity on it? So he said to Fabien and Vaçhek, well it looks like this is an interesting two week exercise for you (laughter). Use X.509 plus a hash of what you expect to find there, and with this very simple mechanism you can replicate many of the trust structures that you find arising for other purposes and from other mechanisms in publishing.

So that's one of the influences. Another influence is Raj Kailar's logic of accountability. If you were going to devise mechanisms which will check chains of signatures of hashes, then this appears to be the right tool, at least on first inspection. It's basically like BAN but without the freshness. You're asking the question whether somebody *ever* signed something, rather than whether he'd signed something during the current protocol run.

Another thing that has given some input to this is Bruce Christianson and Bean Snook's Distributed Object-based Document Architecture which was also described here about six years ago. This is a system that's a bit like a secure RCS, in which you have a number of elementary documents, which are chapters of a book or pages of source code or whatever, and these are held together by folios, which are structures that contain the necessary pointers or makefiles or whatever. And these are integrity protected using hashes and signatures, and there are various demons which can automatically execute certain publicly agreed security policies – such as: a version may be updated if one user manager and one software engineering manager both agree – so you can have automatic demons which will see to it that an update occurs as soon as the relevant signatures appear somewhere in the universe.

And finally, of course, we have to be buzzword compliant, and so you're supposed to do things like adding new security tags using XML (extensible

mark-up language), and the exciting thing about this is that it opens up the possibility of constructing integrity policies incrementally from cascading style sheets. So you can start off with a concept of a signature, and then by saying what there must be in the certificate, you get from that a certificate, and then a bill of exchange might consist of certain items, plus the certificate plus the signature, and so by progressive refinement you can build up the primitives in your business model. So this is where it's all coming from.

Let's be rude about the Orange book. Everybody's rude about the Orange book, I think that's a mandatory part of any talk involving security protocol models. We've got these well known rules of read-down and not up and write-up and not down, and they provide a form of confidentiality, but they don't provide any integrity because a virus can be written up; and they don't provide any availability because a virus can overwrite stuff. Now the full real implementations try to include safeguards against destructive write-up, but you are up against the decision problem, you cannot detect all viruses, this is a theorem from 1983, something like that. So Orange Book doesn't give us what we want for publishing, where we're interested in integrity and availability, but we're not interested in confidentiality at all. In our universe, all files are always world-readable.

If you look at academic research it's 90% on confidentiality, 9% on authenticity and integrity and 1% on availability. And this is exactly the reverse of expenditures in commercial firms, where you may spend many millions on your disaster recovery site, many hundreds of thousands a year on your internal auditors, and essentially nothing on encryption. Can we use Biba? Well Biba doesn't do anything useful for us, because if you touch anything contaminated, you become contaminated to that extent, so if you're running a Biba-type model you certainly can't connect to the net, so that's no use to anybody. Policies such as the Clark-Wilson policy with the BMA policy are useful in specific applications, but they're not really fundamental enough for our purposes. If I had time I might look at Clark-Wilson in this context but it's too much of a manifesto and not enough of something clear and tractable that you can reason about. So, anyway, that's the situation we're in, and what can we do about it?

OK, this is how it should have been done, this is our security policy model and the simplest version of it works like this: each user has one file to which he can append, all files are world readable and nothing ever gets deleted.

Stewart Lee: Thats not a policy, it's a mechanism.

Reply: I consider metaphysics to be the subject of a different lecture course.

Stewart Lee: Well, I think you're wrong.

Reply: So we can assume, for the sake of simplicity, that everything is linear. If you've got a more complex data structure or some little database thingy, then you construct it using the usual mechanisms with pointers and such like. Signatures: I sign anything by simply including it in my file and saying "I assert X". I write that in my file, it's there forever and the whole world can see it, so it conforms with the legal definition of an advanced electronic signature according to the European Union Directive. So everything that I say, unless I qualify it

somehow, I am in some sense signing. And I can countersign something by saying "I assert Bob's document number 4."

Now, what does this tell you? Well, Alice might have been imprudent and simply asserted that she agrees with Bob's document number 4 before Bob ever wrote this to his file, and although that's stupid, stupid people *are* stupid. So you can refine things somewhat by saying that if you want to be sure that you're signing something in somebody else's file and that that is going to work the way it should work, then you want the thing that you point to to contain some kind of nonce, some kind of unpredictability. And you then cite that unpredictability in your own citation, either by reciting it or by stating some hash of it or whatever. You'll see some details shortly.

Now if you do this then you can see that here, for example, Bob has cited Charlie and Alice has cited Bob and Charlie has cited Alice so Lamport time comes out and all these things are working. Now it is of course possible to bring in a number of different kinds of optimisation in this model, but the interesting thing about starting off with such a simple conceptual model is that you can see clearly where you're bringing in an optimisation. For example, it might be possible to decree that the underlying mechanisms would write a true copy of the time along with everything that we wrote with the file, and then Lamport time would be unnecessary for most purposes.

Bob Morris: Is it true that the only thing you can do is append to the file, is that all you can do.

Reply: You can read everybody's file, you can append to your own. But you can only append. You can't delete from the file, not allowed.

Virgil Gligor: So if you make a mistake, it will be there forever.

Reply: Yes, as in company books. Books are written in ink, if there is a mistake you draw a line down the side and you write an update, you produce a transaction that is the equal and opposite of the bogus transaction and that still lies there in the file.

Virgil Gligor: So you have no opportunity to change your mind about something. Never.

Bruce Christianson: That's a different question. You have no opportunity to go back and rewrite an earlier part of your log. The question of what your present position is, that's an abstraction from the log, and you can change your mind about that.

Bob Morris: The pointer into my file is absolutely permanent and unique.

Reply: Yes.

Virgil Gligor: Scary, very scary.

Reply: No, this is real life.

Stewart Lee: Well it isn't anymore, it used to be when you and I worked in an accounting office.

Matt Blaze: But there are mechanisms for building file systems up.

Reply: This is done work. CD-roms appeared what, ten years ago, and people came up with various file systems whereby the directory would be incrementally updated, and then every so often they'd write a new copy of it and so on.

Bob Morris: This is a pain.

Bruce Christianson: Bob's right that this is a real pain if you don't have some software to run on top of the thing to give you some different virtual access.

Reply: Yes, yes, but it's software that runs *on top* of it, that's the point.

Matt Blaze: Disc drives are a real pain if you don't have some software on top.

Reply: Exactly so. Well it's not any worse.

Virgil Gligor: If you insist that this is above the level of the disc drive then I would agree, but I don't think it's far above.

Reply: It's an interesting conceptual model, like Bell-LaPadula, because it tells you when you're optimising, what assumptions you're making when you're optimising, what the risks are and what the costs are. But as I said, it is realistic, banks keep records for six years, OK they put them onto microfiche after six months or so, but that's just mechanism. The bank files are for practical purposes append only. So are medical records, your CV, your laptop if you use it in the way I use mine.

Michael Roe: CVs I thought were a bad example in there, for the rest of them I would agree with you.

Reply: If regulations didn't say three pages only or whatever, then yes. But there are many mechanisms which enable you to construct an append-only file system on top of a bog-standard file system. IBM systems-managed storage, for example, is what banks used to do this. Revision control systems, what we use internally in the Lab to see to it that when, for example, Markus and I write a paper, there is a version that is append-only and can be referred to afterwards if one wants to reverse out changes or whatever. It's all there, it's all fielded technology. Micro payments can be used with disk-writes, if you want to really worry hard about the denial of service aspects then look, we have these mechanisms.

Roger remarked once that when he was Head of Department there were some lawyers came to the Lab, and said that a particular file had to be deleted because the material therein was offensive and belonged to somebody else, and the response was, this is a physical impossibility. But it will vanish in a number of weeks anyway, when it falls out of the bottom of our backup system.

OK, example application, Wei Dai's B-cash. This is another interesting intellectual idea which helps us simplify stuff. B-cash has the idea that a bank is implemented as a newsgroup or a mailing list, if I want to pay Virgil $12.99 then I just post in this newsgroup a message saying, "I hereby pay Virgil $12.99 signed Ross". And that is a declaratory speech act which in itself constitutes the transfer of funds. And everybody who participates in this system is presumed to read all the messages in the newsgroup, and to faithfully execute a programme which calculates how much balance each person has available at any one time. It's got an interesting self-policing property, that if you don't do your sums properly, and you accept a payment for $12.99 from me when I've only got 3 cents in my account, that's your tough luck, you're the person who loses.

Now there are one or two tweaks. You need some kind of secure serialisation, but this can be dealt with and it can all be implemented very, very easily in the append-only world. The optimisation you might use here to do the serialisation would be one file to which everybody could write, so that we've got the serialisation automatically, but again this is an optimisation which you can see and understand and you can compare with the relative costs of, say, Lamport time plus many people writing in their own files. The interesting thing is that this is how banks actually work, because during the on-line day the bank has a number of terminals, ATMs, cheque sorters and so on, who generate transactions which get bundled up together, and you get one journal log for all ATM transactions, one journal log for all cheque transactions, and so on. And at the end of the on-line day, these then get merge-sorted into the account master file using an overnight batch process. This again is an optimisation. It's one that is well understood, and what it's doing, it's simplifying the computation in B-cash, simply by recalculating the initial state in a way that's publicly approved and open to challenge if something goes wrong, through the mechanism of bank statements.

So here's an example of how our security policy model fits in with the real world and enables us to understand and simplify an important business process. So to the extent that the bank's function is mechanical, you can replace it with suitable software, maybe proof-carrying code or whatever.

Frank Stajano: You said that the transaction consisted of you uttering that you would pay $12.99 to Virgil, and then you also said that if Virgil accepted when you've only got 3 cents in your account then it's his tough luck. So obviously the transaction must also consist of some extra sentence that Virgil must have to say, "I think I got that" or something.

Reply: No, if Virgil thinks he has more money in his account than everybody else thinks he has, and he then tries to utter a cheque for more money than he has to somebody else, then that other person will say drop dead.

Michael Roe: Because they run through the list of messages, think a-ha, at this point that was a dud cheque and Ross never got the money so Ross therefore doesn't have the money.

Bob Morris: That's overnight, that's not immediate.

Frank Stajano: When does Virgil get the chance to complain about this cheque for $12.99.

Bruce Christianson: Well he doesn't need to, because everybody can see that it was a dud cheque. So it's as if it never happened.

James Malcolm: But presumably it's only a dud cheque until somebody gets paid.

Bruce Christianson: Well that's why you keep the serialisation, then it's a dud cheque forever.

Reply: Banking systems have many little redundancies built in, in case various aspects of this process go wrong, but that shouldn't stop us understanding what the underlying process of an error-free system would look like, because in that way we can understand and perhaps get some costing assessment of the effectiveness of the different optimisations people have.

Another example that I'm just going to zoom through is, if you look through the various rules of the Clark-Wilson model, then there's some that you can throw away as being motherhood and apple pie, the mutterings about needing a security manager, needing to authenticate users, this is boilerplate, we don't care about it. A number of things that one has in this world – such as a transformation procedure – can be certified valid, we can say that we deal with them in the same sort of way, because our world is an open source for them. Everything is readable including programs, and so it's up to the community to check that transformation procedures, that they use to short-cut various business processes, actually work. Alternatively and equivalently everybody writes their own code. And when one looks at this in a certain amount of detail, our model is doing the same kind of thing that Clark-Wilson does, but it's very much clearer and more concise and more tractable.

Clark-Wilson appears in a number of instances with a number of rewrites, and a number of different sortings of the rules, and it's still very hard to get to grips with what some of the rules actually mean. I summarise here by saying that Clark-Wilson is a manifesto rather than a model.

That's something that I could spend all the rest of the available time arguing about so I won't. What I'm going to do is go on to authentication, and say that in a small system you don't need authentication because everybody reads everything. However, in a bigger system you may want some kind of protocol, providing you access to the attention of other users, or to do some exchange that might be considered to be the equivalent of registered mail. So Alice might say to Bob, Bob here's some content X and here's the nonce, and then Bob might say to Alice, OK Alice, here's some other content and here's a hash of the content that you sent plus a nonce.

Bob Morris: Am I ever going to meet Alice? I've been waiting for years.

Bruce Christianson: How will you know if you do? (laughter).

Reply: So we've got an authentication protocol and we want to verify it, and the obvious first rule to reach for is Raj Kailar's logic of accountability, which is basically BAN without freshness. We found an interesting weakness in this, I'm not going to say out and out that it's a mistake, but in the Kailar logic you can't distinguish between different instances of a receipt or an authorisation response or whatever. There's no mechanism to handle freshness at all, and this becomes apparent in the verification of the Internet billing server which appears in Kailar's Oakland 1995 paper. And there's an ambiguity that's also in the underlying IBS documentation about whether there should be a serial number in receipts or not. Now in the NetBill protocol which followed on from IBS they write everything out in much more detail, and they do have serial numbers, so one may be disposed to give them the benefit of the doubt. However the alleged verification of IBS which appears in the Oakland paper really does skate over the issue of whether, when you see five separate instances of a receipt, you know that five different lots of goods have been received, or whether TCP/IP has been having a bad hair day.

Virgil Gligor: Kailar's logic is supposed to augment BAN, not to displace it. It's not a subset of BAN, so you couldn't say that it had fewer features than BAN. It doesn't do freshness, but that doesn't mean that it's a subset. It's solving an entirely different problem.

Reply: I thought it was plain in the paper that it was more or less a subset of BAN, but with less features than BAN which is its attractive point. But this difference in perception needn't be resolved here.

So, one ends up having to add something or other, and as we've been looking at this the thing that it would appear we have to add is, A said X having seen Y. In other words, there needs to be a concept of temporal succession, and something that lets us get at this temporal succession by looking at the order in which things were said, and whether particular things were unique or unexpected or both. And so the nonce verification rules becomes something like: A said (X,Y) and Y unique means that A said X having seen Y.

Mark Lomas: What do you mean by unique. It's a bit pattern that can't be copied?

Reply: It's a bit pattern that hasn't existed before. At this level all I'm doing is deconstructing "fresh" into "unpredictable" and "unique".

So you see, something like: A said X having seen Y and X unpredictable, means that X was after Y, otherwise you'd have a problem whereby stupid Alice can countersign a document that Bob has not yet uttered.

A serial number doesn't work. Serial numbers are unique but I might know that Bob, being a bank, being given to predictable behaviour, will consume number 125327 in his fourth message of the day, and if I as Alice say, I hereby countersign Bob's bank message for serial 125327, this could happen without her having seen it.

Michael Roe: The standard decomposition of freshness that has been done before, on additions to the BAN logic, has been to separate the concept of an unpredictable random number from a non-repeating number. And there are some protocols where you need unpredictability and some where you just need non-repeating numbers.

Bob Morris: That's brilliant. Unpredictable, that's not necessarily random.

Reply: And you can now see the bug in this protocol here, which is that you actually have to replace it with this, you have to say the hash first and then Y, and we didn't see that until we'd written the logic. Hey, it happens.

Anyway here's the motivation from the Jikzi system. You've got a number of systems out there where people are trying to add things like hashes and signatures to HTML documents. ERL is something we did for medical books, SDML does IBM's things for electronic cheques, DSIG, is the world-wide web consortium's idea for signing content-rating labels, and there's various electronic copyright management systems, and so on. Now these are all application-specific to some extent, and they ignore the needs of some large users like the Department of Defense which will presumably want per-paragraph security labels.

So what we really need is a general enough system to be able to implement different policies, and to be able to deal with inheritance between different types

of object. As I mentioned before, you can start off with a signature, you need signatures and a bill of exchange together with certificates, an insurance contract might contain signatures and certificates and for its effect it might require to have appended to it a bill of exchange and a bill of lading, and then other things might come in before you can actually take delivery of your container of goods. So this is the sort of business model that one gets, with various refinements and various requirements, and one needs a language in which this can be specified. And in order to get it right you need some underlying conceptual model of what we're trying to do and what integrity actually means, and this is what we're trying to provide in the Jikzi system.

What does it look like? There is an early version of the system up on the machines that one can play with, there are various files which state what the structure of a particular document should be, these are the XPF files, there's a type checker, there's a parser, there's the usual things that you'd imagine such as storage and a policy manager and so on. And if you want to see what the code looks like, here's a cheque, and you see you've got all sorts of things such as pay, forename, surname, payment amount, currency, issue date, not-later year, blah, blah. This is straightforward, this is out of STML, it can be obvious to anybody how this sort of thing should be designed to emulate a paper cheque, it all ends up in XML with something that looks like this.

So, last slide. What are the aims and scope of the project. Well, in the real world things go bump in the night: systems fail, files disappear, programs stop working, etc. So a prudent user will keep the data that she needs for her application. But that's not enough. You've got to keep the style sheets, you've got to keep the programs, you've got to keep a large number of things. The US Census I believe still has got the 1920s Hollerith tabulating machine room at the Census Bureau, and the 1930s machinery and the 1940s machinery.

So how can you go about figuring out how much you need to keep? How can you track all the stuff that you rely on? This is what we're trying to do, because we've got the theoretical model which says, in an ideal Universe everybody's got a kind of append-only file, in a real Universe bits of that fall out, so you need to be able to trace and track the stuff that you need. And if there are things like checkpointing going on, then an understanding of that can become clear as well. But the abstract policy model isn't enough, you need a real system so you can implement stuff and write test programs – things like insurance contracts – and see if it works.

Scientifically what we're trying to do is to understand application-level trust structures, not just separation of duty as a mantra that gets uttered in Clark-Wilson, but what are real business models and how do you go about implementing them. And I think it's about time that we made an effort to understand systems without confidentiality, because confidentiality is a pain. Confidentiality complicates everything dreadfully. In most realised systems you don't need it, and in the medium-term future it may only be a relatively small number of organisations that have got it. And if Scott McNealy's prediction is going to come true – namely "privacy, we ain't got none, get over iti" – then why should

we bust a gut and invest a lot of time and effort on confidentiality mechanisms that we personally are not going to benefit from.

The practical side of this is, we want to know how to do electronic publishing in a general enough way that we can deal, not just with academic and medical books, but things like catalogues, software, public key certificates, Computer Supported Cooperative Working, and so on. And there's a server that you can go and play with, and you can e-mail Jong-Hyeon to become a beta-tester.

Wenbo Mao: Where do you put your private key for the application.

Reply: Don't have one. What do I need a private key for?

Virgil Gligor: He said there is no confidentiality, you can't keep your private keys private.

Roger Needham: He never signs anything because there's a mysterious mechanism outside the system that says he can't append to somebody else's file.

Michael Roe: Which is probably actually implemented by a public key cryptography, but that's OK, we understand these levels of abstraction.

Reply: But the point is, that is an integrity restriction, it's not a secrecy condition. I can't write to your file, and so it's an integrity property.

Bob Morris: I have two comments, at least one of them relevent. You should learn about warehouse receipts in your model, you should not deliver things, you should deliver warehouse receipts to turn the whole thing into a paper exercise, so you don't have truck loads of cocoa wanting to arrive in your e-mail. Warehouse receipts are very important, that was a relevent comment. The other comment is that in the banking system you've ignored or omitted interbank transfers, which are handled quite differently from cheques. They go through a different route, they depend on each other in a different way and they're simply handled differently by the whole banking system.

Reply: I started with hierarchical B-cash schemes. I can implement this as well, because if you look at how the actual settlement happens in the interbank system, the head of the clearing system sends a fax to the Bank of England saying the plusses and minuses from the following thirteen banks in the clearing system for yesterday were as follows. In principle that's how it works, the thirteen banks write cheques to or from the Bank of England.

Virgil Gligor: David Reed about 20 years ago had a file system which was multiversion, you could not necessarily delete old versions, and it had interesting serializability properties and very interesting recovery properties, it was one of those few systems that combined concurrency and recovery. I was wondering, how is your idea here different from that.

Reply: I've no idea, I've not read his paper. But the concept of a file system that's append-only is an intellectual model for simplifying and understanding what one actually tries to construct, because what you actually construct tends to be a hash tree of bits and pieces of information on which you rely.

Matt Blaze: Have you looked at the question of layering confidentiality on top of this?

Reply: Oh sure, if you want a spook system you just simplify it by saying that Bob and Charlie's systems are world-readable, but Alice's system is only

readable by Alice, or whatever. And then you just end up magically assuming some kind of invisible read that can happen on your file system without your being aware of it, and that's an implementation detail.

Bob Morris: You've got to be really careful that things don't disappear, because if Alice is unreadable you can't necessarily get to me.

Reply: But things can't disappear, nothing is ever deleted. So you don't get this problem you get in multi-level secure logistic systems whereby a group of storemen classifies ten thousand tons of jet fuel as secret and then take it.

Bob Morris: I should have said things become invisible, not disappear.

Reply: Well things can't become invisible, because things can only be read down. They can't be written up, because I can't write to Alice's system. Alice can read my system, so I assume the pull model of Bell-LaPadula rather than the push model. So all the problems vanish, in fact Bell-LaPadula becomes much simpler.

Bob Morris: OK, good, yes.

Power and Permission in Security Systems

Babak Sadighi Firozabadi* and Marek Sergot

Department of Computing, Imperial College of Science, Technology and Medicine
180 Queen's Gate, London SW7 2BZ, UK
{bsf,mjs}@doc.ic.ac.uk

1 Introduction

It is a standard feature of all organisations that designated agents, usually when acting in specific roles, are empowered by the organisation to create specified kinds of states of affairs – as when, for instance, a priest declares a couple as married and thereby makes it so in the eye of the church, or when a head of department assigns one of his subordinates to a particular project, or when an owner transfers ownership, as opposed to mere physical possession, of an item to another entity. This feature of organisations is referred to variously as '(legal) power', '(legal) competence', or '(legal) capacity'. Jones and Sergot [5] use the term *institutionalised power* to emphasise that this is not a feature of legal systems alone but commonplace in all organisations. The neutral term 'institution' is used by them, and other authors, for any kind of formal or informal organisation.

The states of affairs created when a designated agent exercises an institutionalised power have conventional significance or meaning inside the institution, though not necessarily outside it. For example, ownership of an object, in contrast to its physical possession, is not something that can be observed, and it is possible that one institution recognises an instance of ownership whereas another institution does not. Searl [8] distinguishes between what he called *institutional fact* and *brute fact*. An instance of ownership is an institutional fact; possession is a brute fact.

Policies are institutional facts. For example, the security policies of a system hold within that particular system and perhaps not in any other. As pointed out in [5], words such as *authorisation*, *right* and *privilege* have a wide variety of meanings in ordinary usage. They are frequently used in computer system applications and in computer security, but still not in any standard way. Sometimes by a word such as 'right' we mean a permission, sometimes an institutional power, sometimes a combination of the two, and sometimes something else.

In legal theory, the importance of the following distinction has long been recognised. There are three quite different notions:

1. the power to create an institutional fact;
2. the permission to exercise that power;
3. the practical ability (opportunity, know-how) to exercise that power.

* Part of the work was done during visits to Swedish Institute of Computer Science.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 48–53, 2000.

The distinction between permission and institutional power is illustrated in [5] by quoting the following example from [6].

> ...consider the case of a priest of a certain religion who does not have permission, according to instructions issued by the ecclesiastical authorities, to marry two people, only one of whom is of that religion, unless they both promise to bring up the children in that religion. He may nevertheless have the *power* to marry the couple even in the absence of such a promise, in the sense that if he goes ahead and performs the ceremony, it still counts as a valid act of marriage under the rules of the same church even though the priest may be subject to reprimand or more severe penalty for having performed it.

In this case the priest is *empowered* to marry a couple, which is an institutional fact, but at the same time he may not be permitted to do so. It is common, first to empower an agent to create a certain instutional fact, and then separately to impose restrictions on when that agent may exercise his power.

We sketch a simple formal language for expressing permission and power in security policies, and discuss briefly the significance of the distinction in connection with control mechanisms.

2 Detective and Preventative Control Mechanisms

Part of any security system is its control or monitoring mechanism, which has to ensure that agents are behaving according to the specified security policies. One can distinguish between two basic kinds of control mechanism, called in [4] *preventative* and *detective* control mechanisms. A preventative control mechanism prevents an agent from violating the policies, whereas a detective control mechanism does not *guarantee* that violations are prevented, but will ensure that a violation is detected in some reasonable time.

The distinction between the two can be summarised by saying that a preventative control mechanism satisfies the following formal property, whereas a detective control mechanism does not.

$$\vdash \quad \phi \ \rightarrow \ P\,\phi$$

Here ϕ represents a proposition such as "a reads file F". P stands for a standard deontic permission (see e.g. [2] for details on standard deontic logic).

As formulated here, the violation of a policy is modelled as logical inconsistency. It is the function of the control mechanism to prevent violations by blocking actions that would lead to inconsistency. It is possible to make finer distinctions between different kinds of preventative control mechanisms, for example by distinguishing between what is logically possible and what is practically possible, but we shall not do so here.

In contrast to preventative control mechanisms, in detective ones an agent may be able to perform some action or bring about some state of affairs, without having the permission for doing so. The following formula represents such situations.

$$\phi \ \wedge \ \neg P\,\phi$$

For many applications, the prevention of all non-permitted actions is not feasible or even desirable. For example, a bank may not want to impose practical restrictions on the amounts with which its brokers can trade, even if there are policies prescribing such limits. The managers may prefer to allow the possibility of violations, as long as there is a mechanism for detecting and recording them, for example as part of an audit trail.

3 A Formal Framework

The aim of our research is to develop a formal framework for representing and reasoning about access control policies and meta policies governing changes of these policies. The framework is also aimed to support modelling of control mechanisms to enforce specified policies and rules. We give here just a sketch of the main features.

We distinguish between two levels of policies and the control mechanisms for each. The first level contains access control policies specifying what actions agents are permitted and prohibited to perform on the various objects of the system. These access policies are represented using deontic operators for permission P and prohibition $\neg P$, as expressions of the form $P\phi$ and $\neg P\phi$ where ϕ represents a proposition about actions of type "agent a reads file F", "agent a writes file F", and so on. Note that ϕ in these expressions represents a brute fact, but $P\phi$ represents an institutional fact.

The second level is the level of meta policies regulating changes of access control policies. We introduce a new (relativised) operator Pow, such that $Pow_a\ \psi$ says "agent a is empowered to create the institutional fact ψ". The expression $Pow_a\ \psi$ represents an institutional fact.

In this framework, the only means for changing policies are *declarations*, that is to say, illocutionary acts of *declarative* type in the sense of [8,9]. An empowered agent declares that ψ, and by declaring it makes it so; the fact that ψ holds is due to the act of declaration.

We introduce a (relativised) operator $Declares$ such that $Declares_a\ \psi$ stands for "agent a declares that ψ". In such expressions ψ is a proposition representing an institutional fact – it is not meaningful to 'declare' a brute fact. The proposition $Declares_a\ \phi$ itself, however, represents a brute fact.

Declarations are not necessarily successful (effective). The main relationship between $Declares$ and Pow is the following:

$$[\text{DECL}] \vdash\ Declares_a\ \psi\ \wedge\ Pow_a\ \psi\ \rightarrow\ \psi$$

[DECL] expresses the *exercise* of a power to create ψ by designated agent a.

There are some similarities between our framework and the formal calculus for access control given in [1]. That calculus has two main components, a calculus of principals and a modal logic of principals and their statements. There are two (relativised) modal operators:

– $a\ Says\ \phi$,
– $a\ Controls\ \phi$.

The operator *Says* is defined as a normal modal logic (see [2] for properties of such logics). The operator *Controls* is defined in terms of *Says*, as follows[1].

$$a\ Controls\ \phi \stackrel{\text{def}}{=} (a\ Says\ \phi) \rightarrow \phi$$

$a\ Controls\ \phi$ can thus be read as "agent a is believed on ϕ", "agent a is reliable in regard to ϕ", or, as suggested by ABLP, "agent a is trusted on ϕ". In [1] each entry of an access control list (ACL) is given as a statement of the form $a\ Controls\ \phi$. As the authors explain, since such a statement records a server's trust in a on ϕ, if ϕ represents a request (perhaps in an imperative form) to a server by a, then the server will grant it.

One half of the definition of *Controls* is:

$$a\ Says\ \psi \wedge a\ Controls\ \psi \rightarrow \psi$$

which has a clear structural similarity to [DECL] above. However the two logics are different. We are currently investigating how much of the ABLP calculus of principals can be incorporated usefuly into our framework.

For each of the two levels of policies, access and meta policies, there is a need for a control mechanism to enforce them. In both cases, these control mechanisms can be of a detective or a preventative kind.

For the meta policies, the idea is that in a preventative system, the control mechanism blocks any declaration which, if effective, would violate a policy. Suppose, for example, that the owner a of a file F is empowered to permit another agent b to read F, represented by

$$Pow_a\, P\,(b\ reads\ F)$$

and moreover to empower a third party c to permit b to read F:

$$Pow_a\, Pow_c\, P\,(b\ reads\ F)$$

The formal property of preventative control systems in Section 2,

$$[\text{Prev1}] \vdash \quad \phi \rightarrow P\,\phi$$

where now ϕ stands for any brute or institutional fact characterises a kind of preventative control system for meta level policies also. For suppose that we add to the example the following policy

$$\neg P\,(b\ reads\ F)$$

[1] This definition together with properties of *Says* as a nornal modal operator has some very undesirable features e.g., the validity of $a\ Controls\ \phi \wedge a\ Controls\ \psi \rightarrow a\ Controls\ (\phi \wedge \psi)$.

If a attempts to exercise his power to permit b to read F, the declaration is blocked because $Declares_a P\,(b\ reads\ F)$ leads to inconsistency. However, a declaration by a that c is empowered to permit b to read F, $Declares_a\, Pow_c\, P\,(b\ reads\ F)$, is not blocked. But then an attempt by c to exercise *his* power, $Declares_c P\,(b\ reads\ F)$, is again blocked because it leads to inconsistency.

In general, the property [Prev1] implies

$$[\text{Prev2}] \vdash \quad Declares_x\, \psi \ \rightarrow\ (Pow_x\, \psi \ \rightarrow\ P\, \psi)$$

for any institutional fact ψ.

A system with property [Prev2] but without property [Prev1] is a preventative control system for meta level policies but not for access level policies. A slightly stronger preventative control system for meta level policies satisfies the following requirement:

$$[\text{Prev3}] \vdash \quad Pow_x\, \psi \ \rightarrow\ P\, \psi \qquad (\text{any institutional fact } \psi)$$

Of course it is not always desirable to have preventative control even for meta policies. In that case, in a detective system, the conjunction

$$Pow_a\, \psi \ \wedge\ \neg P\, \psi \ \wedge\ Declares_a \psi$$

can be consistent.

4 Delegation and Role

Delegation is discussed by a number of researchers as one of the issues in distributed systems and in particular access control mechanisms for such systems, see e.g. [3,10]. Delegation is usually defined as giving certain *rights* to an agent to act on behalf of the delegator. However, the term 'right' in this context does not always mean permission, but sometimes institutional power. For example, a delegator issues a proxy to a delegatee to sign certain documents on his behalf. The 'right' that is delegated is not (merely) permission to sign the documents, but the power to create particular institutional facts by signing the documents on behalf of the delegator.

A delegation is a creation of a new policy e.g. a new permission or a new institutional power for the delegatee to act on behalf of the delegator. The act of delegating does not change any brute facts, but if successful, it changes some institutional facts. In our framework the mechanism for delegating is declaration.

There are policies in which a delegator is not permitted to delegate; a delegatee may not be permitted to do what is delegated to him; a delegatee may not be permitted to exercise the power delegated to him, and many other possibilities. Our framework is able to express all these cases, and others.

A well-known method for organising and managing access control policies is by using *roles*. In role-based access control [7], a role is defined as an abstract entity relating a set of permissions to a set of users. In our approach a role relates not only a set of permissions, but also a set of powers to a group of users.

5 Further Work

We have sketched the main features of a formal framework and indicated how it may be used for expressing and reasoning about access control policies and meta level policies for changing policies. We are currently investigating whether incorporating features of the ABLP calculus of principals provides a treatment of groups, roles, and aspects of delegation. It is an interesting question whether the framework needs to be extended with other types of illocutionary acts such as assertion and request. It may be that the full significance will not become apparent until a temporal component is added to the framework. This is an extension on which we are currently working. Without the temporal extension we cannot distinguish between, for example, an agent's sharing of a power with another agent on the one hand and transferring it to the other agent on the other. We are also working on more detailed characterisations of what we called here preventative and detective control mechanisms, and the more general question of what it means to implement a security policy.

References

1. M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. In Joan Feigenbaum, editor, *Proceedings of Advances in Cryptology (CRYPTO '91)*, volume 576 of *LNCS*, pages 1–23, Berlin, Germany, Aug. 1992. Springer. 50, 51
2. B.F. Chellas. *Modal Logic - An Introduction.* Cambridge University Press, 1980. 49, 51
3. Bruno Crispo. Delegation of responsibility. In B. Christianson, B. Crispo, William Harbison, and M. Roe, editors, *Security Protocols*, number 1550 in LNCS, pages 118–124, Cambridge, UK, April 1998. Springer. 52
4. B. Sadighi Firozabadi, Y.H. Tan, and R. M. Lee. Formal definitions of fraud. In P. McNamara and H. Prakken, editors, *Norms, Logics and Information Systems - New Studies in Deontic Logic and Computer Science*, pages 275–288. IOS Press, 1999. 49
5. A.J.I. Jones and M.J. Sergot. A formal characterisation of institutionalised power. *Journal IGPL*, 4(3):429–445, June 1996. 48, 49
6. D. Makinson. On the formal representation of right relations. *Journal of Philosophical Logic*, 15:403–425, 1986. 49
7. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role based access control models. *IEEE Computer*, 29(2):38–47, 1996. 52
8. John R. Searl. *Speech Acts.* Cambridge University Press, Cambridge, 1969. 48, 50
9. D. Vanderverken. On the unification of speech act theory and formal semantics. In Philip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, chapter 11, pages 195–220. The MIT Press, Cambridge, Massachusetts, 1990. 50
10. N. Yialelis and M. Sloman. A security framework supporting domain based access control in distributed systems. In *ISOC Symposium on Network and Distributed Systems Security (SNDSS96)*, pages 26 – 39, San Diego, California, Feb. 1996. IEEE Press. 52

Power and Permission in Computer Systems
(Transcript of Discussion)

Babak Sadighi Firozabadi

Department of Computing
Imperial College of Science, Technology and Medicine

What I am going to talk about is power and permission in security systems. This is work by me and my supervisor, Marek Sergot from Imperial College.

In the area of security we usually use words such as right, privilege, and authorisation, but there are several interpretations of these words. Sometimes by authorisation we mean some kind of institutional power, and sometimes we mean permission. These ideas are not new. David Makinson discussed this distinction in a paper[1], that there is a distinction between being permitted to create a fact, and being empowered or having a legal capacity to create a fact.

The example he used is the following. A priest is empowered to marry a couple, but at the same time he may not be permitted to do so when one of them does not belong to the church. However, if he goes on and performs the ceremonial act to marry the couple, then it counts as a valid marriage. So the couple are married even if it wasn't permitted.

In case of practical ability or practical possibility, that's quite clear to us: I have a practical possibility to open this door, but I may be not permitted to do so. But it is different with institutional power, and it seems that sometimes the same word, permission, is used for two different notions.

So what does this have to do with security? In a paper [2] that I wrote with two colleagues, we distinguished between detective and preventative control systems. In preventative control systems, agents are prevented from violating their policies, so that they are not practically able to not behave according to the policies. But in the detective case they can violate their policies but something may happen afterwards, some other actions may be activated because of that.

$$\vdash \phi \ \rightarrow P\phi$$

This formula would be a property for a preventative process: when we make sure that, if somebody is empowered to do something, then he is also permitted to do it.With a detective control system this is not the case, so somebody may be empowered to perform a certain act, but not be permitted. For example, a bank may have some recommendations for its employees on the amount they can trade with, but at the same time it may not prevent them trading with a

[1] On the formal representation of right relations, Journal of Philosophical Logic, Vol 15. 1986

[2] Formal Definitions of Fraud (B. Sadighi Firozabadi, Y-H Tan and R. M. Lee), in Norms, Logic and Information Systems — New Studies in Deontic Logic and Computer Science, P. McNamara and H. Prakken (eds.), 1999, IOS Press

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 54–59, 2000.

larger amount. What is important here is that the bank manager may want to make sure that a violation of the recommendations is detected, but perhaps not prevented.

My idea is to extend ABLP logic, Abadi, Burrows, Lampson, Plotkin logic for access control, and the idea is to extend this to be able to express the distinction between power and permission. In that formalism, in that calculus, they only have one operator for expressing these things and it's *controls*, when they say "A controls S", and it is defined as "if A says S then S is true".

Bob Morris: If some priest tells me that I am not married, and some other priest tells me that I am married, can you separate out that kind of thing, it's not quite the same as raining.

Reply: Now you are talking about two different agents performing two different actions, that's a different scenario.

So, I've borrowed some ideas from the speech act theory, in which there is a distinction between a declaration of a fact and an assertion of a fact. A declaration is like, when you declare something then it becomes true *because* of the act of declaration, but assertion is not like that. In case of assertion you may say something which is not true, and it does not become true because of the act of assertion. If an agent declares a fact and he is empowered to declare it, then that becomes true. So his act of declaration will have an effect only if he is empowered. For example, if the manager of a company says to one of his employees, you are fired, and since he is the manager he has the power to do that, then it becomes a fact that the employee is fired. But if someone else, not in a mangerial role, says that, then it will not have any effect. With declaration, the fact becomes true because of the act of declaration, but the truth of a statment and the act of asserting it are independent.

Here, we can express things like creation of a permission e.g., A is empowered to declare that B is permitted to read a certain file. This can be an ACL entry. How to delegate a power is something else e.g., A has a power to declare that B has the power to declare that someone else is permitted to read a certain file.

Michael Roe: But there's a problem if A has the power to declare that. Actually that means A is the person who set the policy and at the same time you've got not permitted, so that's a separate policy, but the right hand part of that is then saying ...

Reply: This can be the case many times, if the person is actually allowed to create new policies but some of these new policies that he is able to create may conflict with other policies that already exist. OK, then the question comes, what happens if he exercises his power? That will be a different scenario. Sometimes, because of his role, he may override all the previous policies, or you may get inconsistencies, I don't know, that can be discussed in the context of a specific application.

Ross Anderson: There's nothing mysterious about this. I have the power to make all my files for all readable, I have not ever exercised this power and don't ever plan to. How is this at all problematic?

Bruce Christianson: It follows from the fact that can doesn't imply ought, that there will be some instances where this is the case.

Michael Roe: I think this is going down as what happens when you exercise your right to set policy, and set a policy that is internally inconsistent, so when you effectively declare $\perp$ (*bottom*) in logic terms, what does that mean and does the logic permit you to declare it? And here it's the case where that's actually what you aim to declare.

Bruce Christianson: But that still doesn't give you permission to do it.

Reply: No, it doesn't. The same thing can happen with a delegation, I mean you delegate some power to someone, then the person has the ability to abuse this power.

Ross Anderson: I wouldn't view this as intention at all. I just see it as meaning that B could give himself the power to read F if he wanted to, it happens as a purely contingent factor at this moment in time that he has not done so. Because that's what the assertion "not read" there actually means, it means as an assertion rather than as a declaration, as an utterly contingent fact at this moment in time but not necessarily at any other, it's not the case that B can read F. It doesn't carry any more baggage than that. So I'm standing in for the security manager of a bank and he says, here is the master password in this envelope, and I have the power to read it, and everything in the bank, but I have not as a matter of contingent fact exercised it. How does this description help, that's the problem.

Virgil Gligor: Just think in terms of what discretionary access controls do. There you have at your discretion the power of screwing things up, to do things that go against your own interests. So there is a class of policies in which you are allowed to do things against your better interest, and this ought to be able to express that, and it does.

Reply: But what I have in mind is more like this. Assume that we want a specification of the internal properties of the system, and we are designing a control system, either for detecting or preventing bad behaviour of the users. Then we want to see, how can the system from one state evolve to a certain other state? Is it possible that people who are not permitted to read certain files, nevertheless, because of certain delegations or certain roles that they have or can activate, are able to give themselves certain permissions that are in conflict with the overall policy. In a complex system that can be quite useful to see: is there any way for agents to abuse their power or give themselves permissions to do so?

Virgil Gligor: Two cases, one is exactly the one that you mentioned, I penetrate a system, I have the ability, and I set permissions to a file I am not allowed to. So that's one case that you handle here, and you want an intrusion detection system to find out. I'm an outsider, I don't have a permission, I have the ability, I changed things. But there is another case, where I do have the permission, I have the ability, and I do something that I'm not supposed to, and you are able to express that as well, in this case I'm an insider. So you are

detecting what insiders do and you are detecting what outsiders have done, that don't have the permission but have the ability. So you are handling both cases.

Bruce Christianson: And the purpose of this isn't to produce a shocking conclusion, it's to give us a way of reasoning about preventative policies, that was the example, as well as about the detection. And the point is, you need to be able to do both in parallel, which is a point we often miss.

Virgil Gligor: Which I think is a very good point.

Reply: I think, I'm not sure, maybe many people don't agree with this, but I think the main situation where the thing is needed or useful is where there is the possibility of violation.

If you prevent people from violating, then there's no need to look backwards at what they have done, you know that they don't have the ability to behave badly, if you restrict their actions. But, if people can violate, or in other words they have the ability to violate the policy, then we need to be able to look backward and see what has happened and who has done what.

Bob Morris: Able to, ought to, permitted to ...

Reply: Yes, I think ought, too, because I am thinking of permission in terms of deontic logic, where you can define ought to in terms of permission or the other way around.

Bruce Christianson: The argument is that an obligation is simply a refusal of permission to abstain. We just don't usually think of being obliged to write an audit file in that way, as saying you don't have permission to not log a transaction.

Reply: But also I wanted the ability to distinguish institutional power from practical capacity, which sometimes seems to be confused with permission.

Ross Anderson: I think this kind of speech-act theory is important and very often overlooked, because where you were talking about things such as certificates that are part of a control structure in a distributed system, then they are declarations which have effect of themselves. And we're also looking at the possible legal situation here in the proposed Electronic Commerce bill, where an individual's signature is based on qualifying certificates. One of the big problems with the proposed Electronic Commerce Bill is a confusion between the idea of *declares* and the idea of *guarantees*, because what you actually want is a guarantee.

Bob Morris: Would you like to spend five minutes talking about the role of auditing in the banking industry?

Ross Anderson: It might be a bit difficult mightn't it. For example, when I walk up to an ATM and I punch in my four letter PIN, who in the world actually knows what that PIN is, who should know and who does know, and how is that protected from people who ought not to know. Auditing in that sense is a very difficult matter.

Mark Lomas: The correct answer to that question *should* be, nobody and nothing. Nobody and nothing should know the PIN, except the owner of the PIN.

Bob Morris: That is correct, what you said is precisely correct, but it is not part of the recognised standard. The bank, or anyone at the bank who know your PIN, raises a legal question that is very difficult to answer because if a dispute comes up .....

Ross Anderson: It's very easy to answer, the bank has more money than you so they will win the law suit.

Bruce Christianson: It depends upon what you mean by auditing, because we often use auditing in the narrow sense of saying, we're going to look and we're going to say yes, that's OK, or we're going to say somebody's done something bad so I'll just cry. Auditing in the wider sense says, we're going to go through it, we're going to work out what's actually happened, and on the basis of that we're going to work out what the system state should be, and then we're going to shove the system or some abstraction of it into the state that it should be in. And that's more like the view of auditing that's being advocated here, the detective role of determining what state it should be in and shoving us back into that state.

Ross Anderson: This also happens in the banking world where things are much more serious with bigger stakes, in the dealing room. The dealer can do whatever he wants, and so real banks have got large rows of box tape machines which record everything the dealer says. So you've got the forensic capability after the Barings disaster to go back and retrieve the evidence necessary to put Mr Leeson in jail.

Michael Roe: The distinction between asserts and declares is a very important one that isn't often made, and it raises itself in the context of certificates: when some CA signs a data structure that contains an identifier and a key, saying this other physical entity really is the thing possessing that key, is that an assert, that might be true and might be false, or is it a declare?

Reply: I make a distinction between trust and being empowered. If a person who says something, is empowered to declare it, then whether I trust or not, what he declares becomes the fact, and it has nothing to do with my belief. But assert is more like this: if I trust a certification authority, then I believe what it tells me. But if it has the power to make it the case that a key belongs to a certain person, then that does not have anything to do with my trust.

Virgil Gligor: Is there any notion of trust in *your* logic? The reason why I'm asking the question, I found using the Lampson-Abadi-Wobber logic, and the ABLP logic, that you always need some notion of trust to be able to start your system. In other words I have to say, I trust the following statements, to be able to start the game rolling, otherwise I can't do very much with the logics.

Also you need to distinguish between *trust* and *controls*, because the notion of trust in the other logics is different from the notion of controls. Let me rephrase, if there is no difference between trust and controls you already have the notion of trust. The point is that some entity may control a particular set of actions when that entity is *not* trusted with a different, perhaps intersecting, set of actions.

Reply: I would like to extend this work. The goal is to get to where I can distinguish how permission and being trusted are related, so instead of having

one operator, *controls*, refine this into at least three different interpretations: being trusted on something, being empowered to do something, and being permitted to do something. In a preventative system you don't need to distinguish between being empowered and being permitted.

Ross Anderson: At previous of these workshops, Bob has been educating us to the fact that trust and power are absolutely co-extensive, that a trusted component of a system is one that can break my security policy. If I have a manager who's allowed to sign drafts for one million pounds, and there are two books of 50 bank drafts sitting in the safe to which the manager has the key, then I trust that manager for 100 million pounds, even if he's not authorised to spend anything more than 50 thousand pounds without balancing entries in the branch ledgers, that's too bad, I trust him for 100 million and that's the end of it, because that's what he can nail me for. So there is no difference at all between trust and control in this view of the universe.

Bob Morris: That's really the sentence I wanted to hear, thank you. Is it true that if I am trusted I can make it rain outside, by simply saying it's raining outside.

Michael Roe: But you might be able to make people act in the way they should only act when it's raining. That linkage between trust and control exists.

Bob Morris: Suppose I stand up here and publicly declare it's not raining outside. It is difficult to assign a meaning to that statement [since the blinds are drawn], but it can be done. But the important thing is, if I ask: does anyone here trust me, I will get no response. I should get no response, that has nothing to do however, as far as I know, with whether it's raining or not.

Stewart Lee: I don't understand the connection between it's raining and whether we trust you.

Bob Morris: The connection is not with "it's raining", its with "I said it's raining", and people should start off by saying, *why* did he say it was raining, then I think you'd be starting down the path that would be rewarding, you will get more sensible answers if you take that approach.

David Wheeler: The answer is, we trust you to make misleading statements. On occasions.

Reply: By the way, the banking system makes a distinction between your privileges or your abilities given your role, and entitlements associated with the role. For example, as a loan officer, you have the ability to make loans, but you are only entitled to make loans up to a certain limit. That is a fundamental aspect of access control in banking. Let's make the distinction that Bob talks about in other words, banking does make that distinction.

Bruce Christianson: But the interesting thing when we analyse a protocol is one step back from that, when I'm saying: now, is it that
$<<$ I believe that $<$ this is true $>>$ because $<$ Bob says it's true $>>$,
or is it that
$<$ I believe that $<<$ this is true $>$ because $<$ Bob says it's true $>>>$.
That's where a lot of protocol analyses come unstuck, on that thought.

Stewart Lee: Also the notion of trust can be conditional.

Auditing against Impossible Abstractions
(Transcript of Discussion)

Bruce Christianson

University of Hertfordshire

I want to start with some thoughts that popped into my mind after Babak's talk.

The obvious approach – which I am going to claim is not the right one – is shown in Figure 1. You have, in your model of your system, the set of all the *possible* states that you think it could get in to, based on what people can actually do. And then you've got a subset of those, that you think are the *permissible* states, which are the ones that you're happy about the system being in. And the auditors are happy with them, the shareholders are happy with them, but then there's these other states that you might get into but don't want to be in. And you've got an outer protection perimeter round the possible states, you're saying you can't get outside that outer perimeter but you might get outside this inner perimeter round the permissible states, and you want to protect yourself when you do, and then you've got some sort of audit function.

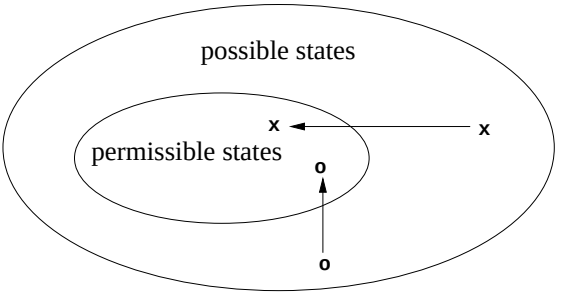

Fig. 1. The Vanilla Model

By an audit function I mean audit in the large sense, not the narrow sense where you just say there's something wrong, but in the broad sense of auditing where you say there's something wrong and you'd better do something about it and here's some suggestions about what you should do.

So you discover that you've done an electronic transaction with someone who isn't going to pay and you say, oh bother, undo the transaction, blacklist the customer, write it off as a bad debt to get the books to balance and the effect of that is that you're now back into a permissible state. So I'm thinking of an audit

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 60–64, 2000.

function as some sort of mapping from the possible states into the permissible states.

Alternatively, given that there are several different states which could get mapped into the same end-state by the audit function, you can think of the audit function as simply giving you a view of the same system at two different levels of abstraction. So in Figure 2 we see the system with all the states that you could be in, I'm going to call them microstates. You view microstates as being *equivalent* if the audit function maps them to the *same* microstate, and I'm going to call these equivalence classes macrostates. And in that case the audit function is simply the specification of a virtual machine, it's a specification of a way of looking at your system to give you only permissible macrostates.

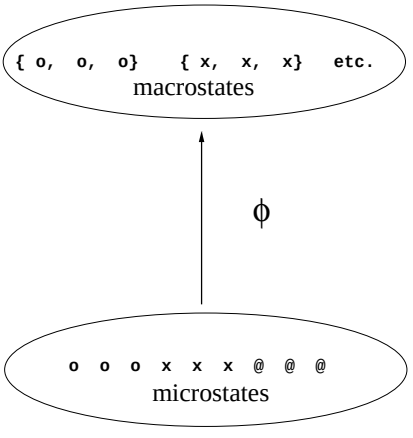

Fig. 2. Two Different Levels of Abstraction

That's the plain vanilla model which is not yet useful or interesting. Adding more flavours, auditing with respect to more than one policy, we now have two different abstractions, two different virtual machines. I should say, that this is all very similar to the view advocated by Tim Gleeson in his thesis about ten years ago [1].

In Figure 3 you have the set of possible microstates, the set of – if you like – actual states that the system could be in at that level of abstraction, possible states; you have the set of permissible states according to the auditing policy ϕ; and you have the set of permissible states according to the auditing policy ψ. In each case you have some way in which the auditing function will pull you from a possible but unpermitted state into a permitted state which we now think of as being a congruence class. And the key point here is, that if you

[1] T.J. Gleeson, 1989, *Aspects of Abstraction in Computing*, PhD thesis, University of Cambridge.

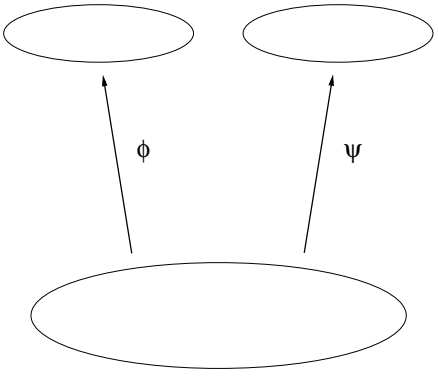

Fig. 3. Two Different Abstractions

have two different auditing policies, then you can discover that your microstate evolutions, your transactions on microstates, aren't congruent with respect to the macrostates anymore, in other words, the microstate evolutions break the boundaries of macrostates.

James Malcolm: Why don't you get sub-state explosion, because if you go from microstate to microstate there'll be more and more states?

Reply: Well, you're going from microstate to microstate anyway, and the macrostates are equivalence classes of microstates so there's fewer of them.

Stewart Lee: A better answer is that this is an abstraction.

Reply: But what the observer sees is that the evolution of the virtual system up here becomes non-deterministic *because* of the presence of the other auditing policy. And I'm not saying that this solves anything, I'm just saying that this a point of view for looking at things which I don't think would have occurred to me before Babak's talk.

Bob Morris: Does that mean you don't know what's happened?

Reply: It means that knowledge of the state you're in and of the transaction you are doing isn't sufficient to tell you what state you'll be in afterwards, and that may or may not be acceptable. So there's two things that pop up from that. The first is, it's clear that it's microstates that have to be archived if you're keeping an audit trail. You've got to archive microstates not macrostates, and since systems are abstractions at all these levels, this means that in some sense auditing is impossible or at the very least you are having to do some pretty heavy compromises with your auditing policy. And there's the second question of whether your auditing policy could actually be a many-to-many mapping on macrostates. Again, these are simply new ways of considering old questions, but I'm suggesting that perhaps this might be a fruitful approach to looking at some of these things.

Ross Anderson: What's happening with the Jikzi model is that we're providing a mechanism whereby you can track all the microstates that matter.

Reply: That's your objective, yes, that's exactly right, and you're providing a number of virtual machines to run on top of that.

Ross Anderson: Yes, and the virtual machine may say, I being the bank, hereby declare that Bob Morris' balance for the purpose of the universe at the end of March is $1,234, and any arguments will be settled by further transactions posted to this account.

So then this becomes the abstraction for value, to decrease the number of gigabytes you have to keep on your laptop.

Reply: I suggest that's quite a good way of looking at what Jikzi does.

But now I want to go back and take issue with the initial model, and this is picking up on points that Rafi made last year to do with optimistic trust or just-in-case trust. You have an *a priori* mechanism for deciding whether you want to engage in a transaction with someone, but Rafi made the point that in practice you really don't want to insist on that too rigidly, because most of the time you want to do this with people you don't know, and you're willing to take a risk that things don't go as they should. So I'm going to argue that the auditing function actually should be extended in both its domain and its range – as shown in Figure 4.

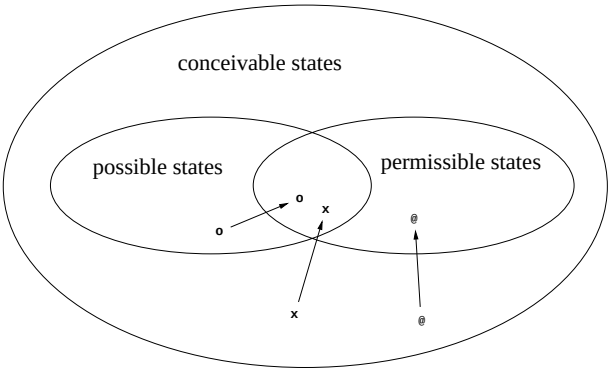

Fig. 4. Extending the domain and range of ϕ

We shouldn't assume that the permissible states are a subset of the possible states. And we should define the auditing function on impossible states, states which we think are conceivable in the sense that it's in some sense physically possible for the system to be in those states, but they're not possible according to our security policy. We do not believe there's any way the system could actually get into an impossible state because of the *a priori* prevention mechanisms that we have. But, nevertheless, there's an advantage to legislating what the auditing mapping does on those states that we can't get into. And the argument for that, the reason for doing that, is that it allows us to take large amounts of the system out of the users' trust envelope. In particular, things like the access

control manager, lock keepers, credit controllers, and so forth, don't have to be in the users' trust envelope anymore. We still rely upon them, I'm not saying that that means they don't have to work, or that we don't have to have them. The user is still relying upon the fact that these things do what they're supposed to do 99.9% of the time. The user may well say, I can't cope if the access control manager gives the wrong answer more than once a week, but the fact that it has given the worng answer once doesn't mean that my system is now unusable forever and I have to throw it away.

The point that I'm making here is that the auditing function on one of these conceivable but impossible states may very well shove you into a permissible state that isn't actually possible either.

Bob Morris: Surely the auditing function is not a function, in that it is not single-valued.

Reply: Well, that is the question on my last slide. If I take an abstract view then I can always ensure that the audit mapping has a single image, I can ensure that it isn't many-to-many by changing the state abstraction. But I think some of those changes, as I said, may have catastrophic effects on the kind of policy that you can audit against. I'm worried about that, it makes me uneasy.

Bob Morris: That is what I was concerned about.

Stewart Lee: I think I understand what you're saying, let me ask: is a conceivable state different than a possible state, or are you describing your own limitations?

Reply: What I'm saying is, that in real systems there are lots of different vested interests, there are users, there are service providers, there's the person who built the system infrastructure, and they have different views about what's possible and what isn't and they'll probably have different auditing policies. That's the case I'm looking at, and I am trying to give a description of how the non-determinism sneaks in, and I'm trying to repair my earlier answer to the objection Stew made about composition of policies.

Stewart Lee: I have made no objections (laughter) I made merely observations from time to time which are very personal. I like this approach though.

What Is Authentication?

(Transcript of Discussion)

Dieter Gollmann

Microsoft Research Ltd.

I thought that this will probably be a free for all anyway, but at least I should try to pretend that I've addressed the particular themes of the workshop. I'm going to twist it because obviously I want to talk about what I want to talk about, and not a topic set by someone else.

If you want to audit protocols you ought to know what protocols are doing. Which leads me to my favourite topic – language for talking about protocols – and leads me to my other topic, where I've given a few talks, including one not so long ago here in Cambridge – authentication. My ideas, observations on this topic have changed a bit, or have been extended a bit since my last talk.

So it is a request for comment, not a solution in the IETF sense. I'll throw out some ideas and I'd be happy to get your opinion. The goal is to get to a better taxonomy for authentication.

Last week I gatecrashed a workshop on coding theory and cryptography at Royal Holloway, where Mike Walker gave a talk on authentication. Mike Walker is from Vodafone, and he really wanted to give a talk on authentication codes. For motivation he started off explaining how authentication works in GSM, a topic I have seen before, several times.[1] You have a user, you have a visited network [VLR] where the user wants to make a call, and you have the authentication centre [HLR]. So the user comes to the visited network and tells who he or she is. The visited network goes to the authentication centre and requests a pair: a random challenge and the random challenge encrypted under the key which is shared by the user's handset. The visited network sends the challenge, the user replies, the visited network compares the two values. Now the user has been authenticated.

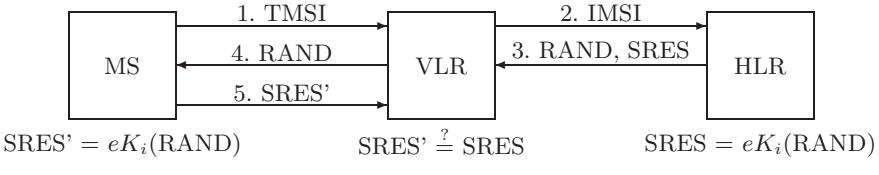

Fig. 1. Subscriber identity authentication in GSM

[1] European Telecommunications Standards Institute, Recommendation GSM 03.20

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 65–72, 2000.

As a matter of language, authenticated by whom? The visited network doesn't know who the user is. Mike Walker's example was "I'm a Vodafone customer and I'm making a call in France". France Telecom doesn't know who Mike Walker is. The authentication centre doesn't make a decision, so who's authenticating?

You could say well, the system. Draw the system boundary larger. Throw in both the visited network and the authentication centre, and then you have your system. I think that is not a good idea because you are lumping things together which make sense being kept apart. And it's the same things that ought to be kept apart that appear in Matt Blaze's talk.

I give you a few more observations. Look at GSSAPI – they talk about security contexts. Look at SSL – they talk about session states. Look at IPSec – you hear about security associations. Look at your typical operating systems – you have something which once upon a time, far before I was competent to talk about this topic, was called a process descriptor segment.

In all cases you establish something which I will call generically a security context and then you receive messages and check that the message belongs to the security context. It happens with GSSAPI, it happens with the Internet communications protocols, and it happens in operating systems. If a user logs in and types in a user name and password, what the operating system does is create a process that will deal with the requests coming from the terminal the user is sitting at. The user makes further requests coming from the terminal and they're evaluated against the privileges described in the process, as specified in the process descriptor segment.

So, my first proposal. If you look at authentication, you have two processes, one is the establishing of a security context, and that's the SSL handshake, that's probably what the authentication protocols do in GSSAPI, that's what you're doing when you log in and type in user name and password; you create a security context. And you have a second step when you check that individual messages belong to a particular security context. And again I was extremely happy about Matt's talk. I like to keep these apart, they're different exercises. His argument was that the first step was something that can take time, hopefully it's done properly and with thought. Checking messages has to be done quickly.

There is a third point that one sometimes sees or reads associated with authentication: the assumption that the user really ought to be a human being. At the back of all of it, at the back of the messages, at the back of the security context, there is really a person. You'll find sources where they really want authentication to mean identify the real human being. That I would describe as a step whereby you associate an entity outside the technical system with something in your technical systems.

You have your system, you have a sender and a receiver, the receiver has the security context, the security context contains a user identity or more precisely an identifier. A context identifier could be a session key. You have a message from the sender to the receiver. You can check that the message belongs to a particular security context (1), and I think it's reasonable to call this data

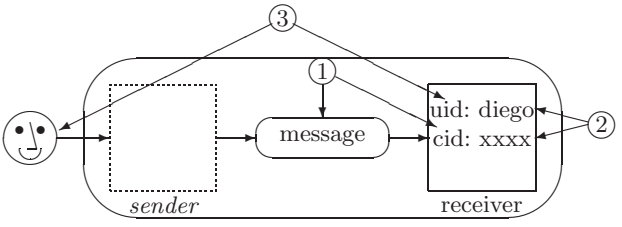

Fig. 2. A new framework for authentication

origin authentication or message origin authentication or origin authentication. You have the process of establishing a security context where, for example, you might want to say a particular session key belongs to particular user identity (2). In the world of cryptographic protocols that might be called key establishment, that's why I call it context establishment. There might be much more interesting information about the security context than simply a user identity.

And there might be a process by which the receiver associates this user identity with a particular person outside the system (3). My first attempt was to call it personal authentication, to make it clear I'm talking about a person. But that's a nasty word, or that's a nasty expression of two words. I wanted something shorter, more elegant. Thinking about the description of certification authorities – where people go to a certification authority and get a certificate binding their public key to their identity – I know that sounds very much a task of the certification authority to create this link, so why not call it certification? To associate an entity outside one system boundary with an entity within the system boundary, call that certification. As I said, this is a request for comment. Better names are welcome because obviously I'm over-using certification in different places.

Maybe most interesting at that stage in my mind is that a security context is something local to a receiver. I'm not associating the message with a sender, I'm associating the message with something that I've set up myself. That's what computers are doing.

Next point, if you scan the literature on entity authentication you will find, starting around 1991, comments that mutual authentication is something different from unilateral authentication going both ways.

The only comment I'm going to make about these arguments is that I don't agree with them and I don't agree with them because they're imprecise in the language employed. One particular interesting piece of taxonomy: you say that a protocol correctly achieves authentication if whenever an agent A accepts the identity of another agent B, it must be the case that B believes that he's been running the protocol with A, which I think is very strange. To check whether you're really talking to B, you have to check what B is thinking.

Bruce Christianson: That's mutual authentication?

Reply: Whatever, you have other definitions of mutual authentication, saying something like, the verifier sends a challenge, the responder sends a response and the verfier verifies the challenge – which is not the direct equivalent.

Bruce Christianson: But B's got to believe he's been running the protocol with someone.

Reply: I could be standing up here and talking to you, not having yet made clear what I think about authentication. You know my identity, but you don't know my beliefs yet. Similarly someone in the audience whom I don't know might make a comment, then I would have that person's belief, but I still wouldn't know their personal identity.

Frank Stajano: It might not even be his beliefs, it might just be what he's saying.

Reply: Anyway, I think it is interesting, slightly peculiar for A to authenticate B and then to get B's state. Now you could go on and remember that in the BAN logic of authentication you sometimes find arguments about beliefs.

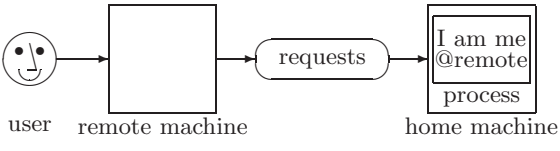

Fig. 3. Source and target authentication

Two terms that I have invented just before. You have a security context at the receiver's side and the receiver can validate, corroborate, authenticate the values in this security context, that's what authentication should mean for me. However, you've also a situation, a meaningful situation where the sender might want to check that the security context the receiver has set up is of the nature that the sender wants. In a financial home banking application, the bank might want to check that the user has set up the bank's software on their PC. So the bank goes, so to say, to the user side to check whether the security context in the user side is appropriate. The names I have proposed here are provisional and I'm again open to proposals – source authentication because you are authenticating the source of the messages you're going to get, and target authentication because you have authenticated the security context at the target that you are going to send the messages to.

Ross Anderson: For source you mean sender not receiver.

Reply: By source I mean that the receiver checks the security context it has set up to receive messages from a certain source. The receiver checks its own security context.

Stewart Lee: But why did you change sender and receiver to source and target?

Reply: Because here [right] I want the receiver to check the security context on the receiver's side and here [left] I want the sender to check the security context on the receiver's side.

Stewart Lee: I understand that, I just wondered why you changed the words, from source to target.

Michael Roe: Rather than have receiver authentication and sender authentication.

Bruce Christianson: Well they're not dual.

Reply: Yes, that's the asymmetry. That's the first time I found a reasonable argument why I want to have an asymmetric situation. I'm talking here about one individual message, I'm not talking about a session where several messages are being exchanged. I'm talking about a situation where, in this particular scenario in the diagram a message is going from left to right. There was a security context at the right hand side, local to the receiver for that message, and that security context should be associated with the message. One situation is where the receiver establishes this security context, and you have another situation where the other party that sends messages to that receiver checks that what this receiver is doing is alright. So you have an asymmetric situation here and I wanted an asymmetric situation, it just brings up two different possibilities.

Michael Roe: I feel very unhappy about that second case, it doesn't really seem to correspond ...

Reply: It corresponds to key confirmation where you want to make sure that the other party got the key you think the other party ought to have.

Virgil Gligor: Well basically the fundamental difference is that if you consider the key to be the only authenticating entity then mutual authentication is running twice unilateral authentication. On the other hand that may or may not be the case if you consider the history of the protocol. What Dieter is pointing out is that he doesn't want just a key to be the authenticating entity, he wants to make sure that the association between the key and the entity at both ends should be known to the other party, that's what he means by key confirmation.

Michael Roe: The service I think is most useful is "I believe that you believe that I sent the message." I sent it to you, and there's a matter of did you actually accept it, and there is a service that convinces me that you are convinced by the messages that I sent to you.

Virgil Gligor: This is one point of view. There is an equally valid, I would say, point of view in which the mutual authentication protocol has a history of the messages exchanged to be able to keep track of beliefs. If you have authentication with a history of messages exchanged, then mutual authentication is no longer twice unilateral authentication, and you could very easily get into a "religious war" on this issue. I still have the e-mail transcripts of a "religious war" that I had with Butler Lampson in March 1994, which basically came down to this: if one thinks the key is the channel, which he did, and authenticated channels is all one is interested in, then these two are equivalent. If you take the other point of view that you are interested in the history of the beliefs and how they have been built up, to be able to say that the two parties have the same history of beliefs,

then that's not equivalent. That point is made by the way by the IBM folks who designed Kryptonite, showing that matching protocol histories is necessary, or arguing it is necessary.

Reply: Arguing in a not very convincing fashion.

Virgil Gligor: In a not very convincing fashion. (And I lost the "war" in '94). Maybe Larry Paulson can tell us about matching histories in logic verification?

Ross Anderson: But the histories are different if you're using a system of key control vectors. Sending key and receiving key are different animals and they come to you by means of different processes.

Virgil Gligor: Possibly, but we're generally talking about the standard mutual authentication protocol where you're actually using the same key. Now which side you believe is really a matter of emphasis.

Reply: There are possibilities you might want to have. Again, in the example the sender is the bank, the receiver is the home PC. The bank, before really starting its home banking application with the customer wants to check that this bloody fool has properly set up his or her security context, is using strong encryption, has the same session key as the bank, not some arbitrary default value which was set to all zeros in a test run.

Michael Roe: But you can't implement that service, because if your assumption is that the client the other end might be broken in arbitrary ways ...

Reply: If you assume the client is broken in arbitrary ways you can't assume anything.

Michael Roe: ... so no protocol could work. On the other hand if you're allowed to assume that the client is OK and then the client is OK.

Reply: Last slide – another take on authentication. That has to do with Martín Abadi's paper. It was last year at the Computer Security Foundations Workshop, where he talked about authentication for credit and authentication for responsibility. The way I want to look at it, you have an actual sender A, sending a message to receiver B, who associates this message with an apparent sender Ã. Who cares if the actual sender is the apparent sender? I give you three options. The actual sender can complain and that is Martín's scenario, authentication for credit. If you want to get credit for the message you're sending and someone else gets credit, you would be unhappy. The other party won't complain because he gets credit. In the second alternative, the apparent sender is unhappy. That would be Martín's example or Martín's aspect of authentication for responsibility. The actual sender wouldn't mind if someone else gets the blame. The apparent sender will complain because the apparent sender didn't send the message. And the last case which brings me back to the track of this workshop, is the receiver. If the receiver is held responsible for how it deals with access control lists. Think of the receiver as a database containing personal data covered by the Data Protection Act and subject to inspection by the Data Protection Registrar. If the receiver was to run a protocol where the receiver couldn't prove that the apparent sender's identity is the actual sender, then the receiver should be in trouble with the Data Protection Registrar. Therefore I

call it authentication for accountability, that is why I have proposed protocols for all these. And that is all I have to say.

Ross Anderson: I think it's simplistic, because if we go back to your initial example of a home location register sending five triples to a visitor location register which in turn does a ping to a GSM handset, what's actually happening is that the visitor location register says to the home location register, "May I give service under standard terms and conditions to Dieter Gollman?" This is, if you like, an offer to contract in the standard legal sense. So the home location register says, "Yes, for each of the following five triples you may bill to me on his behalf calls of not more than six hours", signed Vodaphone, and that's an acceptance of contract, and this is all perfectly straightforward and not mysterious at all.

Reply: Yes, what's the problem?

Michael Roe: The problem is people who call it an authentication protocol.

Reply: Yes. All these people calling it "A has been authenticated." What's happening is that the visited network resident sets up a context, and it sets up the context because it knows where it will get its money from. Matt's trust management system leads in this direction.

Larry Paulson: I suppose since my name was brought up I ought to say something. I never know how to formalise anything to do with belief. I was only looking at the very simple protocols that try to distribute session keys, just to pick up what the essential properties were.

Roger Needham: Initially we just had a symbol and we didn't often interpret it as anything. This was unfortunate as it meant you couldn't talk. I've often regretted that we thought of the name for it that we did, because people thought we were dealing with people's psychology, all sorts of amazing things like that, which wasn't the point at all. It was a fairly simple symbol which had a meaning given to it by the semantics, just as in some sorts of logic they used the expression "turnstile" as the name of the symbol specifically because it doesn't mean anything. We should have done that.

Virgil Gligor: Yes, because essentially what happened was, if you look at the original paper on the Needham-Schroeder protocol, the beliefs which were established at the end points of the two sides were not symmetric. This gave somebody the idea that there may be a vulnerability in the protocol, which was of course discovered before. Because there is usually an insistence on the two parties having the set of meta-beliefs, a lot of people started thinking that all these authentication protocols should end up with the two parties having the same beliefs. So now if that's the case one has to look at the history of the beliefs and if they're not matching, one could say that one does not have authentication. This is a perfectly reasonable point of view, but perhaps not very practical. One can't really derive a lot of protocols based on it, but it's a very different point of view from the one which says authentication is based on channels established by shared keys. One gets equivalence or difference between authentication notions depending upon which of the two definitions of authentication one uses.

Bruce Christianson: I think in practice when we say A believes X, what we really mean is something like A's threat model doesn't consider circumstances where X is false.

Virgil Gligor: I would say that that's a better explanation.

Ross Anderson: Is it better then to say A asserts X?

Bruce Christianson: It depends again what you mean by asserts. Whether you mean A asserts X expecting B to believe that he thinks X is true.

Ross Anderson: If you move from the use of the word believes to the use of the word asserts, you simply drop the doxastic element out of the universe and you cease to consider whether A thinks he's telling the truth or not. Is this too radical?

Bruce Christianson: I think it creates some new difficulties.

Roger Needham: So what we actually meant was, if A and B execute the program correctly they will finish up with a particular string of symbols – end of story. Because programs don't have beliefs. But it's too clumsy to talk like this.

Relations Between Secrets: The Yahalom Protocol*

Extended Abstract

Lawrence C. Paulson

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

1 Introduction

There is something subtle about the Yahalom protocol. Protocols such as Otway-Rees [3] distribute certificates, signed by a trusted authority. Each principal typically receives a session key packaged with a nonce to ensure freshness. But in Yahalom, principal B receives two certificates. One contains a key but no evidence of freshness, while the other is signed using the same doubtful key. To accept the latter certificate as evidence of freshness for the key requires a convoluted argument. It relies on the secrecy of the nonce Nb, which is encrypted using the very key in question; that it still works is surprising.

The Yahalom protocol is largely of academic interest, but equally awkward protocols have been deployed. Kerberos version IV [1] uses session keys to encrypt other session keys. If one session key is compromised, many others could be lost. Despite this vulnerability, the protocol can be analyzed using essentially the same technique that proves the secrecy of Nb in Yahalom. The idea is to reason formally about the relation that holds between the secrets.

The technique is based upon the inductive method [5], which models a protocol as the set of traces that could arise over time. Agents drawn from an infinite population may engage, playing various roles, in any number of possibly interleaved protocol runs. The formal definition resembles informal protocol notation, but contains additional rules to allow the empty trace, enemy action and accidental security breaches. Properties are proved by induction over this definition. If the inductive argument appears not to hold, one can easily identify the offending rule and the circumstances under which the desired property fails. Then one must generalize the induction formula, prove further lemmas to bridge the gap in the reasoning, or look for a weakness in the protocol.

2 The Yahalom Protocol

This protocol, described by Burrows et al. [2, page 257], distributes a session key Kab to parties A and B with the help of a trusted authentication server. At

* The full version of this paper, including details of the Isabelle proofs, is available [4].

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 73–77, 2000.

the end of a run, each party can be sure that the other was recently present.

$$\begin{array}{ll} 1. & A \to B : A, Na \\ 2. & B \to \mathsf{Server} : B, \{\!|A, Na, Nb|\!\}_{Kb} \\ 3. & \mathsf{Server} \to A : \{\!|B, Kab, Na, Nb|\!\}_{Ka}, \{\!|A, Kab|\!\}_{Kb} \\ 4. & A \to B : \{\!|A, Kab|\!\}_{Kb}, \{\!|Nb|\!\}_{Kab} \end{array}$$

Now we can see in detail why Yahalom is problematical. When B receives the fourth message, he obtains a session key from the certificate $\{\!|A, Kab|\!\}_{Kb}$, but it does not mention Nb and could therefore be a replay of an old message. Freshness evidence comes from $\{\!|Nb|\!\}_{Kab}$, but why should B trust a certificate that is signed with an old, possibly compromised key?

The protocol is correct because Nb is kept secret: only A could have formed $\{\!|Nb|\!\}_{Kab}$, so A associates Kab with the fresh nonce. Proving that Nb remains secret is harder than it looks. In an ideal model, one could prove that Kab always remains secret, and the secrecy of Nb would follow immediately for all runs between uncompromised agents. Such reasoning is faulty. It could 'verify' a version of Yahalom that sent Nb in clear in message 2:

$$2. \quad B \to \mathsf{Server} : B, Nb, \{\!|A, Na|\!\}_{Kb}$$

But this version can be attacked. Suppose an intruder I has managed to crack one of B's old certificates $\{\!|A, K|\!\}_{Kb}$, extracting the session key K. He can then masquerade as A, using Nb to forge message 4:

$$\begin{array}{ll} 1. & I_A \to B : A, Nc \\ 2. & B \to I_{\mathsf{Server}} : B, Nb, \{\!|A, Nc|\!\}_{Kb} \\ 4. & I_A \to B : \{\!|A, K|\!\}_{Kb}, \{\!|Nb|\!\}_{K} \end{array}$$

We must be realistic. Old session keys or nonces do sometimes leak out. The inductive model admits such accidents. In order to analyze the protocol, we must then examine the associations between session keys and nonces in the runs of a Yahalom trace. If a particular key Kab is secret then Nb is secret too, even if other keys or nonces are compromised. The proof is long and detailed—as it must be in a non-trivial model.

3 Formalization of the Protocol

The Isabelle formalization (omitted here) follows the usual conventions of the inductive method. The Oops rule is critical to the discussion below. This rule can hand the triple $\{\!|Na, Nb, Kab|\!\}$ to the spy if the server has used these nonces and key together in a run. Oops is intended to model compromise of the session key by any means. Including the nonces in the message identifies the run associated with the key. With Yahalom, however, the loss of Nb is itself a breach of security. Our guarantee to B will say that—provided the run involving Nb has not been compromised in this way—the session key is both fresh and secure.

Many of the protocol's properties are expressed and proved almost exactly like the analogous properties of Otway-Rees [5]. In particular, we must prove the *session key compromise theorem*. This conditional equation serves as a rewrite rule informally states that the loss of one session key does not compromise other keys.

Later we shall consider the security of Nb, where it appears necessary to prove an analogous but more complicated theorem about nonces. A similar theorem is needed to analyze Kerberos IV, and its corollaries include three special cases of the session key compromise theorem. Since Kerberos encrypts session keys using other session keys, the key compromise theorem does not hold in general [1].

The *session key secrecy theorem* states that the protocol is correct from the server's point of view. The session key given out in step 3 reaches only the agents named in that message. The theorem is proved in the usual way. But this guarantee is not enough for A and B. In order to take advantage of it, they require assurance that the certificates they receive originated in a recent and correct server message. Only then can they trust the session key, for otherwise there could be attacks involving reuse of certificates [5].

The relevant guarantee for A states that if A is uncompromised and $\{\!|B, Kab, Na, Nb|\!\}_{Ka}$ occurs in a trace then that certificate originated with the server. The proof is a trivial induction. The argument is that only the server could have issued the certificate (by inspection of the protocol, if you like). The certificate contains all the information A needs: it confirms the name of the other party in the run, namely B, and the presence of nonce Na assures freshness. So A can trust the session key to be fresh and shared only with B.

4 Proving Guarantees for *B*

Half of B's guarantee resembles A's and is nearly as easy to prove. It states that if B is uncompromised and $\{\!|A, Kab|\!\}_{Kb}$ appears in a trace then the certificate originated in a server message of the form

$$\{\!|B, Kab, Na', Nb'|\!\}_{Ka}, \{\!|A, Kab|\!\}_{Kb}$$

for some Na' and Nb'. While A's certificate mentions the nonces used by the server, B's certificate mentions no nonces and conveys no information about how old it is.

Freshness guarantees can only come from B's other certificate. If $\{\!|Nb|\!\}_{Kab}$ appears in a trace then the server said something of the form

$$\{\!|B', Kab, Na', Nb|\!\}_{Ka}, \{\!|A', Kab|\!\}_{Kb}$$

for some A', B' and Na'—*provided* that Nb is secure from the spy. Thanks to its strong assumption about Nb, the guarantee is not hard to prove. Most of the cases of the induction are trivial. The spy cannot create $\{\!|Nb|\!\}_{Kab}$ because he does not know Nb, and honest agents do not try to. The message 4 case holds by a rather complicated argument.

Combining the guarantees for the two certificates meets B's requirements. The server never issues the same session key twice, so the two server messages involving Kab are identical, fixing the values of A, B and Nb.

5 The Associations between Keys and Nonces

Proving that Nb remains secret may seem little different from proving that Kab remains secret, but one detail makes it much harder. The direct analogue of the session key compromise theorem (§3) fails to hold. It essentially says that session keys are not normally used to encrypt other session keys. One might hope to prove that session keys are not normally used to encrypt nonces, but Yahalom's fourth message does precisely that.

A qualified version of the property does hold. If the server has sent the message $\{\!|B, Kab, Na, Nb'|\!\}_{Ka}$ (thereby associating Nb' with Kab) and $Nb \neq Nb'$ then the loss of Kab compromises at most one nonce, namely Nb'. As usual, proving this theorem by induction requires first generalizing it to an arbitrary set of session keys.

The mechanical proof is fairly hard: its script is nine steps long. Message 4, which encrypts nonces, requires some attention; the guarantee for A is used to show that the server associated Kab with Nb. The Oops case involves similar reasoning, but matters can be arranged such that simplification proves it automatically.

In order to abbreviate some of the assertions, I have introduced a relation symbol for the association of K with Nb by some event in trace evs. We typically must show that $\mathsf{KeyWithNonce}\, K\, Nb\, evs$ fails to hold. It suffices either that the key is fresh or that the server has already associated the key with some other nonce. Such lemmas typically have trivial proofs.

6 Proving Secrecy of *Nb*

We are finally in a position to tackle the secrecy theorem for Nb. Suppose a trace contains an instance of message 2:

$$2.\quad B \rightarrow \mathsf{Server} : B, \{\!|A, Na, Nb|\!\}_{Kb}$$

If A and B are uncompromised then Nb is secure from the spy—*provided* no Oops event involving Na and Nb occurs in the trace. We must exclude Oops events for all possible keys because B does not know which session key will be generated. B's final guarantee also requires this strong assumption.

The proof script is long by Isabelle standards (19 commands). Some cases of the induction require some work. The message 2 case splits into three subcases. The message 3 case concerns a new server message containing nonces Na' and Nb', and we must deal with the possibilities $Nb = Na'$ and $Nb = Nb'$. The message 4 case is harder still: here, a message of the form $\{\!|Nb'|\!\}_K$ is sent by some agent, A', and we must contend with the possibility that $Nb = Nb'$.

The Oops case involves more intricate reasoning. If the Oops event betrays the particular *Nb* in question, then it must involve the particular *Na* also, but the theorem statement assumes such events not to occur. If it betrays a different value of *Nb*, then the loss of the session key is irrelevant. Oops is the only case to require the theorem about nonces elaborately proved in §5; we again see that Oops is essential to a realistic treatment of this type of protocol.

7 Conclusions

Combining these results yields B's overall guarantee for the session key. If B has issued message 2 mentioning A, Na and Nb, and receives $\{\!|A, K|\!\}_{Kb}$ and $\{\!|Nb|\!\}_K$ at the end of a run, and A and B are uncompromised, and no Oops event involving Na and Nb has occurred, then the server has issued a valid instance of message 3 involving the particular A, B, Na, Nb and K. Giving this conclusion to the session key secrecy theorem tells us that K is known only to A and B.

Yahalom is far from being the most complicated protocol to be analyzed formally. But it shows surprising complexity, particularly when examined in a model that formalizes the possibility of accidents. Using Isabelle/HOL, I have proved that the protocol satisfies its main objectives. Yahalom's complexity comes from the relation between the two secrets, *Kab* and *Nb*, which are distributed separately to B. The formal proofs for Yahalom demonstrate how to analyze Kerberos [1] and presumably other protocols involving relationships between secrets.

Acknowledgement

Discussions with Martín Abadi and Roger Needham were helpful. The research was funded by the EPSRC, grants GR/K77051 'Authentication Logics' and GR/K57381 'Mechanizing Temporal Reasoning.'

References

1. Giampaolo Bella and Lawrence C. Paulson. Kerberos version IV: Inductive analysis of the secrecy goals. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, editors, *Computer Security — ESORICS 98*, LNCS 1485, pages 361–375. Springer, 1998. 73, 75, 77
2. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London*, 426:233–271, 1989. 73
3. Dave Otway and Owen Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, January 1987. 73
4. Lawrence C. Paulson. Relations between secrets: Two formal analyses of the Yahalom protocol. *Journal of Computer Security*. in press. 73
5. Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998. 73, 75

The Yahalom Protocol

Lawrence C. Paulson

Computer Laboratory, Univeristy of Cambridge

So I'm talking about protocol verification yet again. There are two formal analyses of Yahalom, one of them is a BAN analysis from the original paper and the other one, of course, is mine. Now there are some reactions to this sort of work that I want to dispose of even before I get away from my title slide. The first is, it was all very well doing a couple of protocols but when is it going to stop? And the point I want to make is that some protocols are different from others in a significant way, and that just because this may be the n-th protocol we can still learn things from it. This particular protocol, although very simple, has features which, I think, cause problems with many other methods of protocol analysis. It's really quite a subtle protocol.

The other thing that a lot of people say is, well this is done already in BAN. Well of course it was, and in fact an interesting feature of this is that the BAN proof of this protocol – which I'll show you in a minute – is very concise, as all BAN proofs are, and in fact it gives you the heart, the most interesting part, of the reasoning very succinctly. But as we know, BAN's analysis is in a very, very simple world and at the level at which I look at this, things are very much more complicated. One wouldn't necessarily expect there to be agreement between them, but it turns out there is quite a lot of agreement between them. But there is much more work to do when you analyse it in more detail, I think you do learn more about it.

Now what's new, or what's different about this protocol? Now I still must say that I'm an amateur in the crypto protocol world. I don't really know of that many protocols. Anyway, a more general point, you know that when working in mathematics often before you can prove the main theorem you have to prove a lemma, and sometimes the lemma is as interesting as the theorem. And the lemma that I have found was necessary in a lot of the proofs I undertook was, in English it basically said, that if you lost a session key it didn't matter that much. Now of course you'd lose the session, but in fact when I was analysing the protocol I never actually modelled the sessions themselves, so they didn't count, but most of the protocols never use a session key to send anything important.

Bob Morris: They send the session.

Reply: Oh yes. Well, I don't care about the session.

Bob Morris: Fine. I do not disagree.

Reply: OK, what I mean by "anything important" is, anything important to the operation of the protocol itself. There are some exceptions. Kerberos is an exception – I think probably the most well known one – because you have session keys that encrypt other session keys, and so the spy gets the session key which she can then use repeatedly to get other sessions, right, and they are encrypted with the first session key. And here you have what looks like a

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 78–84, 2000.

catastrophic loss, which is that if the first session key gets compromised then an unlimited number of other session keys get compromised. And we all know intuitively that somehow these losses end very promptly, because the one session key only controls a certain other session which is timed and eventually it will all stop.

Virgil Gligor: Unless the ticket granting ticket is extensible.

Reply: OK, yes. If it has an unlimited lifetime then it would be worse, because you have an unlimited number of losses. One might still want to prove that the system continued to function even with bad session keys being generated all the time, I think that's feasible.

What's difficult when you try to prove this formally? One likes to have simple results, and this bit of English here, when it's expressed as a formula, it can be expressed relatively succinctly and it's a thing you can use in a natural way and prove. As soon as you give that up, naturally life is harder. And I'm not looking at Kerberos here, though Gianpaolo managed to verify Kerberos, I'm looking at a much simpler protocol, Yahalom, which as far as I know has no practical use but is very good as an example of a lot of things.

Incidentally, you probably know that everybody has taken the Needham-Schroder protocol and beaten it absolutely to death as an example, and one thing I would like to do is see that one retired at last. It's a bit like the stack example that was used in about nine thousand papers on abstract data types, it got completely exhausted. The Yahalom protocol has only got one message more but much, much more is going on. I don't know if I should do this in much more detail, but – you know this already – let's say we do. You've got this nice triangular structure, it's the usual thing, that's trying to distribute the session key. So there's Alice talking to Bob, who then sends something to our authentication server, who sends the tickets back to Alice, and finally she sends things to Bob. And because of the triangular structure, everybody gets involved after the previous person in such a way that they're all meant to believe that the other person had this too. That triangular structure actually has got nothing to do with my proof.

But the particular feature of Yahalom, it's a really perverse feature that wasn't necessary and it makes it interesting. I can now show you a little more carefully what the difficulty is. As you know, in a run of these protocols, Alice and Bob expect at the end to get certificates or tickets that say, here is your key and this is why you should believe it's good. Now the certificate that Alice gets for herself is a reasonable one, because it says here is your session key, and it's encrypted with her shared key, so she might believe it's actually valid, it has the nonce that she chose and it has the name of the person it's shared with. So that's quite a reasonable certificate, it has everything she wants, and indeed when you set it all up formally, one can prove very easily that Alice should be happy.

As for Bob, things are tricky, because for – as I said – no reason that I can think of, the certificate that Bob gets from the authentication server doesn't have his nonce in it. And so Bob gets something that says, there's this key, it came,

we don't know how old it is, so it might be good, but then maybe it's not so good after all. And then the other thing that Bob gets, and this comes from Alice, is the nonce that he chose, encrypted with the same key. Now the BAN analysis of the Yahalom protocol says that this nonce can be a secret, and because it's secret – and it is, sort of – nobody else could have created that message. It must have been Alice who created that message, because our intruder, Linda, doesn't know this nonce.

Now if you look at this, you might find it a little hard to believe that it all holds together. The most doubtful thing, which even complicated it enough using BAN, is that one is reasoning about a certificate that is encrypted with the key that you don't actually trust – see this key – but when you're looking at the certificate, all you know is that you don't know how old the key is, you don't really believe it's our key anyway, and yet somehow you're inferring something from the certificate.

Now what did *I* do? No, I'm not going to go in great detail about the proof method, anyone who hasn't seen my paper I'll be happy to discuss it with you afterwards, but roughly speaking I have mechanised things in a fairly intuitive way. And roughly what I do is, I set up this protocol and in addition I set up a mechanism that ensures that session keys can be lost at any time. In other words, I've taken it that session keys are vulnerable and you shouldn't count them as staying around for terribly long. And, somehow, then try and prove that Bob has reasons to believe in the certificates that he gets. And the point of this whole talk is that, in order to do this proof, I need to use formalisation techniques that are not needed for protocols that do not have this kind of relation between secrets. I should be more explicit: *this* secret is used to encrypt *that* secret, and if this secret becomes compromised – lost – then that secret is also lost. And the question is, can we cope with the protocols that do that kind of thing.

And just to – very blatantly – show you what can go wrong. If the nonce is not secret – if for example we modify the protocol so that in the second message Bob's nonce was sent in clear – then the trivial attack lets Linda now have an old certificate, with an old session key that is no longer secure. She can observe this second message and, because the nonce is in clear, she can use its encryption under the old session key and trivially make a new run of the protocol.

Now the interesting thing is that BAN did this sort of thing very well. Roger's here to contradict me if I'm wrong, but I think BAN was largely set up to deal with authentication for freshness, as opposed to anything else. And when we look at more recent methods such as model checking, they were more concerned with attacks that involved tricky interleaving of two or three runs and splitting things up in crazy ways that one couldn't possibly work out by hand unless perhaps one's very good at solving chess problems. Maybe that's fine here, but most people couldn't find those kind of attacks. And the funny thing is now, looking at some recent work – I'm not going to name any names, in fact I haven't even given hints as to the name I'm thinking of – sometimes they are not sensitive to this kind of attack. This kind of attack is much too simple, it's just a violation of freshness, and one can see people who have claimed to have proved a thing,

when there is a kind of gaping hole in the wall like this, that they've forgotten about because their model assumes that session keys stay secure forever. And the thing that you need to do, after the fact, seems very straightforward – although I can tell you at the time it wasn't particularly easy – we simply formalised the relation between secrets such as the keys and the protocol.

Now this bit here is just reminding you very briefly about the method that I used, but it would take a whole talk really to give any detail on that. I made this new relation symbol, which here I call *key with nonce*, its arguments are a key, a nonce, and this thing that stands for a list of events, it's a trace, the history of everything that's happened since the start of time, everything that's happened on the network. And the only kind of events on my network that I am interested in here are – I know this is illegible but what it means is – those events which consist of message three of the protocol, in which the server sends out a session key and a nonce together. And if the server sends out a message in which he uses this session key together with this nonce N_b then this relation holds, and otherwise it doesn't.

And so our proofs will simply be concerned with proving properties of this relation, and intuitively there's not a lot going on here. For example, if you were to look at the protocol then you would think, the key here, this particular key, can be associated with at most one nonce, and so, for example, this relation is the graph of a partial function, an obvious property. Mathematically I think all the properties being proved here are – I won't say obvious – none of them terribly deep. There are some complicated chains of reasoning in some places, but nothing very exciting.

Just to give you some very, very simple examples to give you a flavour of what goes on here, the relation fails to hold if the key is fresh for a trace, by which I mean the key doesn't actually occur in the trace at all. Clearly the key is fresh, how can it be associated with the nonce? Another case, if the key is associated with some other nonce. As I just said, the key can't be associated with two different nonces, it just doesn't happen in the protocol. Every time the server issues message three, it always chooses a fresh key. It's not in the habit of recycling session keys, so you will never have the same key being used with two different nonces. And that means – and this is an important property – that if the server had associated its key with some nonce N_b' which is different from N_b then bingo, it has not, will not, cannot be associated the key with N_b as well.

Now things of this sort can be proved very easily even by machines. One of the paradoxes, something that really confuses people, is if you prove things mechanically it's actually harder than proving it on paper. The reason is that all the obvious things have suddenly to be explained in excruciating detail. If you're lucky enough the prover will prove those things for you but it doesn't always end that way. So you say, why do we ever use mechanical tools at all, and that's because if you look at the case analysis you get from some of these protocol steps, you find cases that go on for two pages or something like that. You probably are not going to want to simplify those two pages yourself, and so that's when it becomes worthwhile using one of these dumb tools.

Anyway, the main lemma that I proved using this relation – let me translate it into English – is as follows: if the key is associated with some nonce N_b then all the other nonces are completely unaffected by the loss of that session key. And as you see, I can express this as a nice rewrite rule, which is important because most of the proofs are based on rewriting. When I come to a later theorem, the important theorem, remember the important property is that this nonce is actually secret. I use this lemma in the proof that the nonce remains secret and so whenever, in all the different case analyses, the possibility comes up that a certain session key might have been made available to the spy, then by this rewrite rule, unless this nonce was actually the very one that was distributed with that session key, then it doesn't matter whether that session key is compromised or not. And this is tolerably hard to prove, I should say a lot of these things are really very hard, but in fact this was one of the most difficult proofs that I've done.

Now I have an overview of all the proofs that concern Bob. Here are the two certificates for Bob. From the first one we trivially get that the session key was, at least at one point, a good one, it came from the authentication server and so on. What do we have here? Firstly it's very easy to prove as well that, provided this nonce is secret then the session key is fresh. So I can get to this point with really very little effort, the difficulty is showing the secrecy of the nonce. So again, people who say it was all done in BAN before, we must remember that BAN doesn't talk about secrecy, so the hard part to prove hasn't been covered yet. We then need to prove that this nonce is secret, which we do using the lemma I showed you on the previous slide, and using the technique of the relation which I mentioned to you. And then finally by Modus Ponens we obtain that the key is fresh. Now it's a funny thing that the freshness and the goodness or authenticity of the key are proved separately. Knowing that the key is fresh but not knowing who was intended to use it is not much of a guarantee, is it? And there are plenty of attacks in which you get a fresh key which isn't meant for you. So you need the two of these together.

Now one point that I think I can claim is, that this approach is general, or at least it is general in the sense that it works with two different examples, and Yahalom is one, the other one being Kerberos IV. And Giampaolo and I have tried various other ways of verifying Kerberos IV before resorting to this, so it isn't that easy to do, but in the end I see no other way. Now of course it's a slightly different situation, there you have a key encrypting another key so relations will be between two keys and so on, but it turned out we could prove a general lemma which then one could specialise, into I think three different versions, turning on the combinations of different types of session keys that you were worried about losing and so on.

Why are the proofs so hard, how hard were these proofs? I don't remember exactly but I should think the whole analysis might have taken two weeks, which is a significant amount of time. And I think it would not have taken so long if I had known the technique – obviously if I'd have known how to do it – if I knew

that technique it might have not have been quite so bad, but these mechanical proofs always are difficult.

Another point I should admit is, the model I use is perhaps overly pessimistic. To show you what I mean let me go back almost to the very start, here's our protocol again. Now of course we know that session keys cannot be trusted forever, but the way my protocol model said it was, keys can be lost at any time, and it might even allow the session key to be lost as it is going from S to A, already at this point a session key could be lost, bingo, which I think is a little pessimistic. I think people can look after their session keys better than that. But in fact the most complicated cases in several proofs are precisely the case in which, at this particular instant of time, the session key has suddenly been found out, and it may be that if one could somehow have a less pessimistic model then one could simplify the proof just a little, though in fact the structure of the proofs would have to be the same because I do believe in the overview of what this protocol's doing.

Mark Lomas: You say they may have been careless with their keys, I'd like to suggest an alternative. We discussed auditing before, if you're using it purely for authentication you might choose to record it in your audit trail of session information, to allow someone to validate the behaviour of the protocol. So it's not that you're necessarily being careless, you might as a matter of course, in effect publish your session keys.

Reply: OK. Not being really a security person myself, I never understood precisely why people didn't trust a session key, but it clearly has something to do with the numbers of them that there were and so on, clearly their very name gives them that throw-away aspect. But naturally one shouldn't be too careless because you must be using them for something and you don't want *all* your secrets to be lost by mistake.

Mark Lomas: Clearly what I just suggested is not a good idea for encrypted secrets, but if you're merely wanting to authenticate something ...

Bruce Christianson: It might be, for instance, that I'm submitting a document to you for a tender or something like that. The purpose is to ensure that other people can't see it. While they're preparing theirs they can't see it, but it's part of the protocol that the tenders must become public after a certain time, and it must be possible for everyone to verify that the thing, which I'm claiming is the tender I submitted, really is what I sent to you. So that's an instance where revealing the session key would be part of the protocol.

Ross Anderson: That assumes publicly monitored communication channels.

Bruce Christianson: Oh, it assumes all sorts of things, but we're in the context of auditability.

Reply: I never intended that session keys *have* to be published, only that they might be.

Virgil Gligor: Larry, you mentioned that you verified Kerberos 5, for the last slide it was Kerberos 4, but previously you said somebody verified Kerberos 5. The differences are very significant. For example when you say that you are

verifying Kerberos 5, you undoubtedly will say that you verified the forwarding feature and the proxying features.And the forwarding feature in Kerberos has an uncanny ability to make you lose session keys. So I would be curious to see what the results of the application of your logic to the Kerberos 5 forwarding and proxying would be. And forwarding comes in two flavours. One where I can forward to you my identity – and a lot of things about me – within session keys, and the other one is that I forward my identity to you and you can forward it further. I'd like to see how – and whether – you can say anything meaningful in your logic about that. In other words, I suspect that that feature is so weak that you might not be able to say much meaningful about it.

Reply: What one would hope to say – and typically all I tend to say – is that if that goes wrong then you're sunk. All I intend to be concerned with is whether the rest of the world can carry on as normal.

Virgil Gligor: The interesting thing is that part of the rest of the world that's involved in this forwarding. Here the world is very closed and static, but there it becomes dynamic, and "the rest of the world" becomes a reasonably well-defined thing but dynamic.

Reply: Well that sounds interesting. I'm glad, I *thought* there must be many examples in which you have this more delicate relation between secrets.

Virgil Gligor: Very much so.

Modelling Agents' Knowledge Inductively

Giampaolo Bella

Computer Laboratory, University of Cambridge
New Museums Site, Pembroke Street, Cambridge CB2 3QG (UK)
Giampaolo.Bella@cl.cam.ac.uk

Abstract. This note introduces my extensions to Paulson's "Inductive Approach" in order to reason about agents' knowledge. First, I argue that an agent knows all the components of those messages she creates. I exploit this notion on some crypto-protocols with particular message structure. Then, I allow a new event, message reception, to occur on the network. This leads to a broad definition of agents' knowledge, which extends the existing definition of spy's knowledge. I will discuss strengths and weaknesses of the two models when proving session key knowledge.

1 Overview

Paulson's "Inductive Approach" [6] is a powerful framework to reason about the confidentiality goals [4] met by modern cryptographic protocols. The expressiveness of the inductive definitions provides realistic models (agents may conspire with the spy, session keys may be accidentally lost, etc.), while the theorem prover Isabelle supports the mechanisations of the proofs.

The approach relies on the concept of *trace*, which is a list of events that can occur on the network. The protocol model is defined by specifying how to build inductively all possible traces of events, according to the message exchanges required by the real-world protocol. Specifying all possible traces signifies that the model includes all possible situations arising when an infinite population of agents run the protocol. The spy is amongst these agents.

The first considerations about agents' knowledge in the Inductive Approach arose when I formulated the theorems stating confidentiality of session keys in such a way that the agents could verify their assumptions [3]. By invoking these theorems, the agents are able to *learn* that certain session keys are confidential, so increasing their knowledge.

Other approaches allow deeper reasoning about knowledge. For instance, the BAN logic – though criticised about its soundness – contains a predicate of the form "A and B believe that they may use K to communicate", while state-enumeration techniques can enforce the peers' agreement on a set of data [5]. Both approaches allow reasoning about the agents' knowledge of relevant message items.

So, I started to investigate the matter within the Inductive Approach, focusing on the knowledge of session keys. A pair of peers A and B typically consider a key K "good" for their communication when they both have evidence that K

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 85–90, 2000.

is confidential, and that K is *known* to both of them. I will concentrate on the latter requirement.

As a general remark, I must point out that crypto-protocols are conventionally run by a population of agents (also said network parties, or principals), when in fact they are run by the agents' workstations. This difference is rarely pointed out in the literature (e.g. [1]). I will hold to the convention and speak about agents' knowledge instead of workstations' memory cells. I will not deal with the authentication of agents to workstations.

2 Model 1: Hacking with Message Creation

Some of the shared-key protocols analysed inductively so far have a common feature. After the trusted server has created a new session key and the peers have obtained it, each agent uses the session key to encrypt a *new* message, and then sends it to the peer. Although this design could be criticised as an incautious way to achieve authentication, it can be exploited to prove that each agent knows the session key.

One possible strategy to prove this is showing that the agent is the true creator of the message encrypted under the session key. The ability to use a key to build a cipher obviously implies the knowledge of the key.

More formally, an agent A is the creator of a message msg meant for her peer B on some trace evs, when A has sent some message msg', containing msg as a component, to B on evs, and msg never appeared before this event:

$$
\begin{aligned}
&A \ \mathsf{Issues}\ B \ \mathsf{with}\ msg \ \mathsf{on}\ evs \equiv \\
&\quad \exists\, msg' .\ \mathsf{Says}\, A\, B\ msg' \in \mathsf{set}\ evs \ \wedge \ msg \in \mathsf{parts}\{msg'\} \ \wedge \\
&\quad msg \notin \mathsf{parts}(\mathsf{spies}\,(\mathsf{takeWhile}(\lambda x.\, x \neq \mathsf{Says}\, A\, B\ msg')(\mathsf{rev}\ evs)))
\end{aligned}
$$

Recall that $\mathsf{parts}(\mathsf{spies}\ evs)$ yields all message components of all messages in the trace evs, assuming readable ciphers. Traces are always extended from the head, therefore the "rev" function must be applied.

$$
\begin{aligned}
&1.\ A \rightarrow \mathsf{S} : A, B, Na \\
&2.\ \mathsf{S} \rightarrow A : \{\!|Na, B, Kab, \underbrace{\{\!|Kab, A|\!\}_{Kb}}_{ticket}|\!\}_{Ka} \\
&3.\ A \rightarrow B : \underbrace{\{\!|Kab, A|\!\}_{Kb}}_{ticket} \\
&4.\ B \rightarrow A : \{\!|Nb|\!\}_{Kab} \\
&5.\ A \rightarrow B : \{\!|Nb - 1|\!\}_{Kab}
\end{aligned}
$$

Fig. 1. The shared-key Needham-Schroeder protocol

I have proved that the Issues predicate holds on several messages of BAN Kerberos, Kerberos IV, and the shared-key Needham-Schroeder protocol (Fig. 1).

For example, B's knowledge of the session key on the Needham-Schroeder protocol (**Lemma 1**) can be stated as follows: if B sends message 4 of the protocol, i.e. the event

$$\mathsf{Says}\, B\, A\, (\mathsf{Crypt}\, K\, (\mathsf{Nonce}\, Nb)) \tag{1}$$

appears on a trace *evs*, then

$$B\ \mathsf{Issues}\ A\ \mathsf{with}\ (\mathsf{Crypt}\, K\, (\mathsf{Nonce}\, Nb))\ \mathsf{on}\ evs$$

The conclusion assesses that B is the true creator of the cipher, which can be read as B *knows* K. Lemmas of this form usually require a confidentiality assumption on the key K, otherwise the spy could forge the cipher and confute the conclusion. However, in this case B encrypts a fresh nonce so the spy cannot tamper with it.

Clearly, this technique cannot be applied if the peers do not use the session key after its reception. However, the peers do know the session key *when* they receive it. To express this property, I decided to model message reception.

3 Model 2: Defining Agents' Knowledge

The event $\mathsf{Gets}\, A\, msg$ formalises agent A receiving message *msg*. The protocol model allows a message sent by A to B to be possibly (not necessarily for the network is insecure) received by B. The same message can be received more than once, and even by the wrong recipient [2]. Incidentally, this model must be used to formalise non-repudiation protocols and express properties like non-repudiation of reception.

At this stage, I have defined a function knows that takes as parameters an agent and a trace, yielding all messages that the agent has handled on the trace. The base case states that an agent knows her initial state. The inductive steps (the case for the Notes event is omitted here for brevity) state that each agent knows what she sends or receives, while the spy knows all messages ever sent.

$$\mathsf{knows}\, A\, (\mathsf{Says}\, A'\, BX\, \#\, evs) \triangleq \begin{cases} \{X\} \cup \mathsf{knows}\, A\, evs & \text{if } A = A' \ \vee\ A = \mathsf{Spy} \\ \mathsf{knows}\, A\, evs & \text{otherwise} \end{cases}$$

$$\mathsf{knows}\, A\, (\mathsf{Gets}\, A'\, X\, \#\, evs) \triangleq \begin{cases} \{X\} \cup \mathsf{knows}\, A\, evs & \text{if } A = A' \\ \mathsf{knows}\, A\, evs & \text{otherwise} \end{cases}$$

Recall that a message can be received only if it was sent, therefore the inductive step on Gets does not need to enrich the spy's knowledge. This definition extends and replaces the existing definition of the function spies [6].

Agent A's knowledge of a message *msg* is formalised by $msg \in (\mathsf{knows}\, A\, evs)$ for some trace *evs*, while A's knowledge of a message component, say a key K, is expressed as $\mathsf{Key}\, K \in \mathsf{analz}(\mathsf{knows}\, A\, evs)$, meaning that A must extract K from a message amongst those that she knows.

In this model, B's knowledge of the session key on Needham-Schroeder (**Lemma 2**) states that if B receives the "ticket" sealed under his shared key, i.e. the event

$$\mathsf{Gets}\, B\ (\mathsf{Crypt}(\mathsf{shrK}\, B)\, \{\!|\mathsf{Key}\, K, \mathsf{Agent}\, A|\!\}) \tag{2}$$

appears on a trace *evs*, then B can access K from the traffic she has handled,

$$\mathsf{Key}\, K \in \mathsf{analz}(\mathsf{knows}\, B\ evs)$$

4 Proving more Significant Guarantees

At this stage, I want to make the above results available to the respective peers. More concretely, while those lemmas merely tell the agents what themselves know, I will now build up from them more crucial guarantees telling the agents what their peers know. I will still concentrate on knowledge of session keys.

In Model 1, I can easily prove by induction that if

$$\mathsf{Crypt}\ (\mathsf{shrK}\, A)\ \{\!|N, \mathsf{Agent}\, B, \mathsf{Key}\, K, X|\!\} \in \mathsf{parts}(\mathsf{spies}\ evs)$$
$$\mathsf{Crypt}\ K\ (\mathsf{Nonce}\, Nb) \in \mathsf{parts}(\mathsf{spies}\ evs)$$

and K is confidential, then event (1) must be on the trace *evs*. The message $m = \{\!|N, B, K, X|\!\}_{Ka}$ is required to appear in the traffic on *evs* to pinpoint the peers for K. Confidentiality over K is needed to prevent the spy from forging the message $m' = \{\!|Nb|\!\}_K$ (not needed in Lemma 1 because Nb was assumed to be the fresh nonce created in step 4); this assumption can be relaxed to further requirements that A can partially verify [3].

Refining Lemma 1 by this result yields **Theorem 1**: if the messages m and m' appear in the traffic and K is confidential, then B *knows* K. Therefore, If A ever gets hold of m and m' and verifies the confidentiality of K, then she can apply the theorem and deduce her peer's knowledge of K. I have proved a symmetric guarantee for B.

Similarly, in Model 2, if the events

$$\mathsf{Gets}\, A\ (\mathsf{Crypt}\ (\mathsf{shrK}\, A)\ \{\!|N, \mathsf{Agent}\, B, \mathsf{Key}\, K, X|\!\})$$
$$\mathsf{Gets}\, A\ (\mathsf{Crypt}\ K\ (\mathsf{Nonce}\, Nb))$$

occur on a trace *evs*, and K is confidential, I can prove that

$$\mathsf{Says}\, B\, A\, (\mathsf{Crypt}\, K (\mathsf{Nonce}\, Nb))$$

occurs on *evs*, and then that event (2) also occurs on *evs*. Therefore, Lemma 2 can be refined by this result producing **Theorem 2**, which is a guarantee for A: if A receives the messages m and m', and verifies K to be confidential, then B *knows* K.

5 Discussion

Theorems 1 and 2 are practically equivalent because they become applicable to A at the same instance of time, i.e. when she gets hold of the messages m and m' (clearly, A must first decrypt m in order to extract the session key K to decrypt m'). Both theorems establish the same result, which may be interpreted as *B's non-repudiation of knowledge of K with A* because it forbids B to mislead A about B not knowing K.

The only difference is that, since Model 1 does not contain message reception, Theorem 1 must refer to some earlier time: the instance when the messages appeared in the traffic. Consequently, it could be argued that Theorem 1 is somewhat stronger because it enforces its result earlier than Theorem 2 does.

The requirement of confidentiality of the session key is necessary to prove that a certain event involving such key could only occur if some other event did previously occur on the same trace. As described above, this strategy was followed for proving both theorems, which therefore must assume a confidential session key.

Consider a protocol whose last message delivers a session key, say, to B. In this scenario, Model 1 cannot help because no Issues property can be proved. By Model 2, I could still enforce a result of the form of Lemma 2 but not of Theorem 2 because there exists no event that implies B's reception of the last message of the protocol. Therefore, the result of Lemma 2 cannot be made available to A.

Now, think of a protocol which delivers a session key to B and then terminates with B sending a message to A. Even if the message does not contain the session key, I can exploit this last event to establish a result of the form of Theorem 2. Therefore, Model 2 seems to have a broader scope.

The two models could be easily combined to capture a broader notion of knowledge, but I believe that Model 2 can elegantly cope with most realistic circumstances. Both models may be used to reason about knowledge of any message items.

Acknowledgements

Special thanks to Larry Paulson for supervising my Ph.D.

References

1. M. Abadi, M. Burrows, C. Kaufman, B. Lampson. Authentication and Delegation with Smart-cards. DIGITAL Technical Report 67, California, 1990. 86
2. G. Bella. Enhancing the Inductive Approach by Message Reception. *Technical Report No. 460*, Cambridge University, Computer Laboratory, 1999. 87
3. G. Bella. Are Timestamps Worth the Effort? A Formal Treatment. *Technical Report No. 427*, Cambridge University, Computer Laboratory, 1998. 85, 88

4. G. Bella, L. C. Paulson. Kerberos Version IV: Inductive Analysis of the Secrecy Goals. Proc. of *Fifth European Symposium on Research in Computer Security*, Springer, LNCS 1485, 1998. 85
5. G. Lowe. A Hierarchy of Authentication Specifications. Proc. of *Tenth IEEE Computer Security Foundations Workshop*, 1997. 85
6. L. C. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. *Journal of Computer Security*, 6:85-128, 1998. 85, 87

Modelling Agents' Knowledge Inductively
(Transcript of Discussion)

Giampaolo Bella

Computer Laboratory, University of Cambridge

Today I am going to talk about how to model agents' knowledge using the inductive approach, which we will just see in an instant, and we'll talk about formal verification. I think there's probably no need to motivate this type of work. When we talk about agents' knowledge, we means things such as A believes that a session key is good for communicating with B. We need this key to be confidential, we need message integrity, but we also really need to be sure that somehow B knows the same session key so that two parties agree on the same session key and can communicate securely.

I think there is also work done in the field of state enumeration techniques recently by Lowe who has tried to include this feature. I am going to talk about something that we can do with simple inductive approaches. Our models are infinite, we don't have a limited population of agents, we can define the protocol goals and then prove them formally, including temporal aspects. We can prove something like: a particular session key is confidential as long as it has not expired.

The proofs are quite easy to follow because humans doing mathematical proofs often tend to reason using induction. I believe they're quite intuitive although, as Larry pointed out, there are several details that we need a prover, a tool, to deal with. We had certain real-world bench marks: I would say that the method is expensive, which doesn't mean that you've got to pay a lot to buy it, but that you need to use a lot of manpower to really master it.

So let's go deeper. I will show two models. One is based on the approach by Paulson, where we only have a single network event, namely messages. There's no reception, agents can only send messages. I believe a good way to formalise agents' knowledge where all you can do is just send messages, is showing that you can actively use some message to create a new message and send it, and you are the true creator of the message. In other words if I can use some message items and encrypt them using a particular session key, then clearly I must know both the key and the message items, so that's the idea of how to formalise knowledge in this case. Then I'll present another model, which is based on the approach that I have extended by message reception, and here I have a slightly different notion where I say that an agent knows the items that she can retrieve from the messages she can handle. What I mean here is that if I send a message then somehow I see the message, I can handle the message, I know the message. And the same if I receive it, but if I want to talk about knowledge of some message items I must somehow extract these items from the messages that I can handle. That's the main idea.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 91–94, 2000.

So in the first case I simply had to introduce a predicate. I said that on a given trace E, A issues B with a particular message M, so A is the true creator of the message M, meant for B, and M did not appear in the traffic before. M did not appear in the traffic, so that's why we are sure that A is the true creator of M, and therefore A must know all components inside M. In the case that M is a cipher, A must also know the encryption key to synthesise.

Let's see how we can exploit this notion. We're talking about how to model session key knowledge in BAN-Kerberos. What I mean by BAN-Kerberos is the simplified version of Kerberos that we've read in the BAN paper. In this case I can prove the following lemma, that if the session key K is confidential and B has sent encrypted with it this cipher over a certain trace E, then B is the true creator of the cipher, and therefore B must know all components, and in particular must know the session key K.

Let's look at the assumptions. I don't want to talk about the confidentiality assumption, there's other work about it and you can look at it separately if you want. Let's look at this assumption here, this is an event performed by B so B, if everything goes all right, is the only entity on the network that can check that the event occurred, that can verify this assumption. So here the theorem is just telling B what B himself knows, so the importance is merely theoretic, but it is important because I can exploit it to prove something more interesting. I can enforce this event to happen based on another assumption that this time A can prove. So combining these two lemmas, I get the relevant guarantee. And this time I am telling A what B knows, because this says that provided that the key is confidential — there are ways to relax this assumption, as I said, and I won't to go into the details of that — if this message appears on the traffic then B is the true creator of the cipher, and therefore B knows the key. Clearly if A at some point gets hold of this cipher, A infers that the cipher appeared on the network and therefore that this assumption does hold and therefore A can invoke the theorem, guaranteeing to A that B does know the session key K, so A is assured that the session key is all right for communicating with B. B has non-repudiation of knowledge of K, so B cannot ever argue that he doesn't know K.

So that's an example using the first approach, and let's go to the second modelisation. This time we build a set of messages that each agent knows. This is an inductive step, and the base case simply states that what A knows on the empty trace is the initial state of A, of the agent. This is the inductive step of a reception event, this is a concatenation of the trace with an event, and here I'm saying that we can extend the knowledge of A over a given trace E by the message X that is received. This means that each agent knows what the agent alone sent or received over the trace, but the spy basically knows everything any agent could ever send or receive. So, that's a definition of knowledge.

Stewart Lee: This is not a criticism of what you have done, but a more general criticism. I am getting increasingly worried about the fact that we refer to the computers that are communicating in an anthropomorphic way as people. We think of them as she and he, Alice and Bob. We think of them knowing

something when what we really mean is that the data is there such that a human being seeing it could discover it, and that sort of thing. We think of believing in, which is somewhat different than knowing. I'm not sure that this kind of approach, this anthropomorphic approach, doesn't lead us to conclusions that are unjustified in the sense that if they were really were human beings they might know it, or believe it, or whatever, but they're computers, they're machines, and they only know what they're told to know.

Reply: You have a user sitting there in front of the workstation ...

Stewart Lee: It's not the user who has anything to do with what you're talking about.

Virgil Gligor: There is also another problem if you assume that the parties are machines, not humans. Humans may not be able to do what computers do. For example, one of the things that humans typically don't do is to issue three requests in an interval of a millisecond. If you issue three requests within one millisecond which is much faster than a human could do without tools, you can get colliding time-stamps on things which really messes things up.

Bruce Christianson: It's actually stronger than that: if you see a piece of text encrypted under a DES key, it's a pretty good bet that a human didn't do it.

Virgil Gligor: Do these kind of analogies, in both directions, ever cause a problem? This is really the question. Stewart thinks that it might.

Stewart Lee: I'm sure it doesn't cause a problem in the reasoning, but I'm not sure we're reasoning about the right thing.

Dieter Gollman: There's no problem with the reasoning, it's interpreting the results.

Bruce Christianson: Part of the problem is with the pronunciation of these symbols written "knows" and "believes". If we said "renders" and "represents" then we'd probably be making less out of it.

Virgil Gligor: I think that's the problem that Roger pointed out earlier during Dieter's talk about beliefs. I had to believe what Roger said, and if that's the case then the whole thing took us down a path that we didn't want to take. Reaching the same beliefs at the end of an authentication protocol is a requirement that may not be relevant. If you have that as a requirement, I can guarantee that mutual authentication is not twice one-way authentication. That may be an artefact of a semi-historical event which is that misunderstanding or the mislabelling of things. Stewart was right. This makes his point.

Michael Roe: When talking about the BAN logic you always have to make a health warning that "knows" and "believes" are just symbols in the logic that don't mean knows and believes in the human psychological sense. I think we mostly take that for granted.

Virgil Gligor: I'm sorry to report otherwise.

Michael Roe: There is a more subtle way it's misleading. You're assuming a one to one correspondence between a piece of hardware and a person who uses it. That is often swept under the carpet, it's true.

Matt Blaze: I'm just wondering whether or not it gets you into trouble to confuse a computer with the person controlling the computer.

Bruce Christianson: Well it depends if there's only one person in control.

Michael Roe: It isn't the user who writes the software, so when it goes wrong it isn't necessarily the user's fault.

Matt Blaze: My computer committing a fraud is not necessarily the same as my committing a fraud.

Michael Roe: Designing your protocols on the assumption that they are equivalent can lead to trouble.

Bruce Christianson: And the person to whom a computer's actions are attributable is not necessarily the person responsible for those actions being carried out. So you have to do quite careful book-keeping before you can say what the bottom line is, who is responsible.

When we say that A believes that B believes X, we mean something quite formal and complicated[1]. But we don't like saying this, it's awkward on the tongue, so we use these other words.

Matt Blaze: On the other hand I think the trouble with the confusion may be outweighed by how much easier it makes talking about protocols when you have a he and a she, that we refer to as Alice and Bob.

Stewart Lee: I accept that entirely, but we have to keep it in mind.

Bruce Christianson: If you're investing all this effort in agreeing on a session key, it's important to have people who believe it, that there is some hope of a conversation you're going to have. But again there's this danger, that we like words like signature, we use them as metaphors, and all sorts of parameters of that metaphor get copied across that we weren't aware of.

Stewart Lee: Another one is watermark.

Michael Roe: There are protocols where two machines share a symmetric key — from the machine-level view it's authentication — but from the point of view of liability and responsibility, the person who can actually administer the machines is in fact the same at both ends.

Bruce Christianson: When I am logging on to Hatfield from a CCSR machine, I want to make sure that it really is me at the other end.

Stewart Lee: You mean you meet yourself at the other end?

Bruce Christainson: No, but I want to be sure that the resources that I control at the other end are the ones that I think I am in control of when I'm in Hatfield.

---

[1] In symbols, if W_A is the set of worlds (trace prefixes) which are consistent with A's security policy and, for $w \in W_A, \pi_A(w)$ denotes the subsequence of protocol events of w directly observed by A, then "A `believes` X [`after` w]" means: $\forall w' \in W_A : \pi_A(w) = \pi_A(w'), X[\texttt{after}\ \ w']$. Hence "A `believes`. B `believes` X [`after` w]" means: $\forall w' \in W_A : \pi_A(w) = \pi_A(w'), \forall w'' \in W_B : \pi_B(w') = \pi_B(w''), X[\texttt{after}\ \ w'']$.

However, if A and B are in different security domains and may have different security policies, so that $W_A \neq W_b$, then W_B is not the set of worlds consistent with B's *actual* security policy, but with the policy which A *meta-believes* B to be using. And this form of meta-belief need not be transitive. You can run, but you can't hide.

Time-Lock Puzzle with Examinable Evidence of Unlocking Time

Wenbo Mao

Abstract. Rivest et. al. proposed a time-lock puzzle based on the RSA crypto algorithm. To unlock the puzzle a player has to perform a computation which has a well understood time complexity. However, in that puzzle scheme there is no way for a player to examine whether a puzzle has been constructed in good order and to play a puzzle may foolishly risk wasting a very lengthy time on trying to solve an intractable problem. This weakness prohibits the time-lock puzzle from any applications that involve distrusted parties (most imaginable applications do involve distrusted parties). We propose a new RSA-based time-lock puzzle which includes an efficient protocol to let a player examine the time needed for unlocking.

1 A Previous Approach

In reference [4], a time-lock puzzle is proposed which is based on an RSA computational problem.

Suppose Alice has a message M that she wants to encrypt with a time-lock puzzle for T seconds. She generates a composite modulus $n = pq$ with p and q being two large primes. She sets t such that $t = TS$ with S being the number of squarings modulo n per second that can be performed by the solver. Alice then generates a random key K for a conventional cryptosystem, such as RC5. She encrypts M with key K which outputs ciphertext C_M. She now picks a random element a modulo n, and encrypts K as

$$C_K = K + a^e (\mathrm{mod}\, n),$$

where

$$e = 2^t (\mathrm{mod}\, \phi(n)).$$

She finally outputs the time-lock puzzle (n, a, t, C_K, C_M) and erases any other variables created during this computation. With the knowledge of $\phi(n)$, Alice can construct the puzzle efficiently.

Assuming finding M from C_M without the correct key is computationally infeasible, so is computing $\phi(n)$ from n, then it seems the only known way to unlock the puzzle will be through computing $a^{2^t} (\mathrm{mod}\, n)$ which leads to finding K from C_K. Obviously, this computation requires t steps of squaring modulo n.

An important advantage of this puzzle is that the process of repeated squaring is "intrinsically sequential". There seems no obvious way to parallelize it to any large degree. So the time needed to solve a puzzle can be well controlled by Alice.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 95–97, 2000.

However, the scheme has a major drawback: there is no control whatsoever over Alice. A number of things can go wrong with Alice and the result will be an unsolvable puzzle. In usual applications of timed-release crypto (e.g., time-delayed key recovery), the puzzle maker's successful cheating will mean total defeat of the system security. The trouble here is not that we should not trust Alice, it is the absence of any means for Alice to prove her honesty. This problem prohibits the time-lock puzzle from most timed-release crypto applications.

2 A New Time-Lock Puzzle

The new time-lock puzzle proposed here is equipped with a protocol that allows Alice to efficiently prove her correctness in construction of a puzzle. Solution of a puzzle now means successful factoring of a large composite modulus that she has generated in the puzzle. So the new time-lock puzzle is in fact a timed-release factorisation of a large integer. We relate the factoring problem into that of computing a "small" discrete logarithm which has a well understood and controllable time complexity.

We shall only highlight the working principle of the scheme. (Though this is quite sufficient for full understanding of the puzzle.)

Alice shall setup $n = pq$ such that -1 is quadratic non-residue modulo both p and q (so it has the positive Jacobi symbol). Setting $p \equiv q \equiv 3 \,(\mathrm{mod}\, 4)$ will satisfy this condition (n is then a Blum integer [1]). She shall disclose an element modulo n of a hidden order (most elements in the RSA group has a hidden order). Let g be this element and the hidden order be T. This means

$$g^T \equiv 1 \,(\mathrm{mod}\, n).$$

Alice has chosen g such that $g \bmod p$ is a quadratic residue modulo p and $g \bmod q$ is a quadratic non-residue modulo q. Thus, g is a quadratic non-residue modulo n and it has the negative Jacobi symbol (the negative Jacobi symbol can be efficiently checked by the puzzle solver). Knowing p and q permits Alice to find such g easily. Also, algorithmically setting p and q permits her to control the size of the hidden order. (We shall provide the procedure details in a full paper.)

Given g, there exists efficient (non-interactive) protocols to show the size of the secret order T (i.e., the bit size of T in the binary representation). Such a proof is all what Alice needs in order to show her honesty. When T is relatively "small" (e.g., $\leq 2^{100}$), the proof also clearly indicates the time needed for finding the secret order from g, 1 and n. The time will be measured by $t = \sqrt{T}$, the best known algorithms to achieve this: Shank's baby-step-giant-step algorithm [2] (Section 5.4.1), Pollard's catching-kangaroo algorithm [3].

Now suppose that T has been revealed from t steps of computation. Let U be the largest odd factor of T. Then we know that

$$(g^U)^{2^i} \equiv 1 \,(\mathrm{mod}\, n),$$

for some i. Note that U is odd and g has the negative Jacobi symbol, so $g^U \pmod{n}$ still has the negative Jacobi symbol. Recall that -1 has the positive Jacobi symbol, and so does 1 (easy to check), therefore

$$g^U \not\equiv \pm 1 \,(\mathrm{mod}\, n).$$

To this end we can conclude that $g^U \,(\mathrm{mod}\, n)$ is a non-trivial square-root of 1. Thus

$$gcd(g^U \bmod n \pm 1,\ \ n) = p \text{ or } q.$$

The time-lock puzzle is solved within an unlocking time which is set under the control of an examinable evidence.

References

1. Blum, M. Coin flipping by telephone: a protocol for solving impossible problems. Proceedings of 24th IEEE Computer Conference (CompCon), 1982. pp. 133–137. 96
2. Cohen, H. *A Course in Computational Algebraic Number Theory.* Springer-Verlag Graduate Texts in Mathematics 138 (1993). 96
3. Pollard, J.M. Monte Carlo method for index computation (mod p), *Mth. Comp.*, Vol.32, No.143 (1978), pp. 918-924. 96
4. Rivest, R.L., A. Shamir and D.A. Wagner. Time-lock puzzles and timed-release crypto. Manuscript. Available at (http://theory.lcs.mit.edu/~rivest/RivestShamirWagner-timelock.ps). 95

Time-Lock Puzzle with Examinable Evidence of Unlocking Time

(Transcript of Discussion)

Wenbo Mao

Good afternoon. I'll be sticking to the auditability theme. This is about a protocol which was proposed by Rivest, Shamir and Wagner. It's a timelock puzzle and to start with I will look at what is a timelock puzzle and what is its use, and then look at the RSW scheme, and then it will be an obvious requirement for auditability, to establish that the puzzle can really be solved within the stated time.

Before going into details I look at what the puzzle actually is. It is timed-release cryptography, which takes a very long time, or any specifiable length of time, to solve. Once it's solved you then know some bits of crypto. It is based on RSA, and now there is argument about how to name RSA, somebody says it's the alleged trademark of the cryptography used, so somebody else says it's *secret order*, like *discrete logarithm*, address the problem rather than the inventor's name.

Now the applications of time-release cryptography. Obviously there are several, say a bidder wants to seal a bid for a bidding period, another thing is sending messages to the future, a secret to be read in 50 years' time, and another thing is key escrow architecture. Key escrow is this thing where there is a requirement to escrow some keys so that they can be recovered, and the danger is vast scale intrusion. So with timed-release cryptography it will take some time to produce a key, although we mustn't waste a tremendous amount of time, but vast scale penetration becomes infeasible, becomes an individual criminal does not have the resources, so this is an example of a real application.

Now look at the RSW scheme. It is based on a secret order to an element. Suppose Alice has a secret to encrypt with a timelock puzzle for t units of time to solve. She generates two big primes p, q and multiplies them to obtain n, and then picks a random session key K and encrypts with this the message M using conventional key cryptography to get C_M. Then she encrypts the session key K using RSA, by adding a^e modulo n to give C_K. Here a is a random element and this exponent e is defined as $2^t \bmod \phi(n)$ where t is the number of timesteps needed to solve the puzzle. Since Alice generated p and q she can compute this e easily, whereas without knowing the factorization you cannot compute $\phi(n)$. Now C_M and a and C_K are published, so this triple becomes the timelock puzzle. So if we analyse it we know that to decrypt message M from C_M you need obviously the correct key, assume this, and to decrypt K from C_K you need to compute $a^e \bmod n$. Without knowing the factorization of n it seems that the only known way to compute a^e is by a repeated squaring of a, so that is t multiplications.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 98–102, 2000.

This procedure is intrinsically sequential, since there is no way to parallelise it. No matter how you compute, you may have any tool, it will actually still get the same number of operations, so it's sequential.

Michael Roe: There's some fine-grained parallelism in computing a square, because if I write the number out in a power series and then two of them are multiplied together, it looks like I can do in parallel at least some of those multiplies. We've still got the additions which I can't parallelise but I can do a lot of the multiplication, so I can gain a fair bit on squaring on a parallel machine. I think you're right in the sense that there is a limiting part through the middle of it that can't be parallelised but I can gain about $\log n$ by having parallel processes that work on different parts of it. Beyond that I don't get any speed up at all, so as long as t is much bigger than $\log n$ then, yes, intrinsically sequential.

Bruce Christianson: The argument is that the time is *eventually* linear in t.

Matt Blaze: It seems that it may be non parallelizable for one instance, but somebody who wants to parallelise a large number of these can certainly do a large number in parallel.

Reply: OK, but for one instance you cannot do it, so that's not an attack to this model.

Virgil Gligor: So we are treating here the problem of time release, I want to release, for example a key, at a certain time, and to make sure the bits are not opened earlier than a certain date. Can I do that as a matter of the protocol as opposed to as a matter of encryption? There is a protocol which is presented at the annual Computer Security Applications Conference last December by Michiharu Kudo from IBM Tokyo Research Lab on a time release protocol, and this paper was purely based on classic public key cryptography, there was absolutely nothing new in there. So that gives the impression that you could do more or less what you do with this new sort of encryption scheme without this kind of encryption.

Reply: Oh, yes, I guess that's true, this is only one way.

Virgil Gligor: So perhaps it would be good to think about the differences between solving the problem with conventional cryptography as a matter of a protocol, or solving it with a new form of crypto.

Ross Anderson: There's a small point there, it's perhaps not prudent to say that the time is linear in t. There's a nice point that Serge Vaudenay made, that if you ask a really naive first year undergraduate how much longer it will take to search 128 bits of key compared with 64 bits of key, then you expect the student will say twice as long. Now given Moore's law, this is actually correct because if you're given $1M and 20 years to do key search, you do nothing at all for 19 years and six months and then you do the whole thing, and so your key search power is up exponentially and so it does take you twice as long to search 128 bits than 64. So you have to be mildly careful, I think, not to say

that it's ultimately linear in t, unless of course you put in another "two to the power of"[1].

Reply: OK, next slide. Before starting to solve the puzzle, you know that t is big and it takes an awful deal of time. Before doing that, you want to know that it is a good puzzle: can I really get a good message, or will I just waste the time t? And in this scheme there isn't any way to be sure, you just have to trust the puzzle-maker. And we know that that's a problem.

Actually, we know that release even part of the secret and we have the power to do factorization, one quarter of the length of n and we can factorize. So for RSA it's a big problem, not so straightforward.

So this work here is to look at how to add-on some way to gradually release the secret, without causing sudden factorization, until the secret is released enough. So this is adding a protocol to this scheme, because as I said it will allow Alice to prove her honesty in making the lock.

So again it is based on RSA, so Alice should start out, set up with $n = pq$ choosen so as to make –1 have positive Jacobi symbol. This is very easy to ensure using Blum integers, so that $p, q \equiv 3 \pmod 4$, and this is a standard way although not really recommended but it's not harmful. And as we know, most elements in RSA have a secret order, Alice can select an element g with hidden order T so that $g^T \equiv 1 \pmod n$. Alice publishes g and keeps this order T secret. And we know 1 and –1 both have positive Jacobi symbol and we know $g^T \bmod n = 1$ and so has positive Jacobi symbol and we know g has negative Jacobi symbol, therefore T must be even.

Now knowing p and q permits Alice to choose such a g by algorithm[2], and by setting p and q Alice can also choose the size of T to be large, but anyway this is the trick, how to set n.

Now Alice wants to give a proof, and prove the size of T without disclosing T and this can very efficiently done, it can be done with a protocol and it's basically a proof of the number of bits in T. Alice has published g and n and provided the number of bits in T is relatively small, less than 100 bits, it will be the case that T can be found in in the time complexity of the discrete log algorithm. It's well known that this can be done using $\sqrt{T}$ number of multiplications, so if we have a number we can set that up between T and $2T$ with that precise level of control, so such a proof is all that is needed to enforce the protocol for partial key escrow.

How to do this is based on extracting a small discrete log. So we now look at how it works, we know that T is even, and suppose U is the largest odd factor of T. If we set $h = g^U \bmod n$, we know that $1 = g^T \bmod n$, we also know that both +1 and –1 have positive Jacobi symbol, and g has negative Jacobi symbol, and U is odd, therefore h has negative Jacobi symbol, and h^2 has positive Jacobi

[1] If p, q are Blum integers then $\phi(\phi(n)) = 3\phi(n)/4$.

[2] We ensure that g is a quadratic residue mod$p$ but a non-residue mod$q$, whereas –1 is a quadratic non-residue modulo both p and q.

symbol; therefore h cannot equal +1 or –1, but we know[3] that therefore $h^2 \equiv 1 \pmod{n}$.

So we have $(h+1)(h-1) \equiv 0 \pmod{n}$. Since h is not +1 or –1, –1 is just $n-1$, so these factors $h+1, h-1$ must both be bigger than zero and less than n, and their product is a multiple of n, so actually p and q must divide one factor each. So the greatest common divisor of $h-1$ and n is p or q. Now this is your controllable way to prove that n can be factored within a specified time.

So finally I will open for discussion, for a proper setting size T, the square root $\sqrt{T}$ can be a reasonably big size for various applications. When T is bigger than 20 bits the square root $\sqrt{T}$ will be 2^{10}, which is about 1K steps, so you can choose a set-up of T which can be decrypted within minutes or which can be decrypted within 50 years.

I am going to assume any algorithm to extract small discrete logs also will not parallelise well, we know the optimum so far, we know — unless we have some breakthrough along the way — using M processors only achieves $\sqrt{M}$ speedup. This is not very efficient for parallelising. And finally I think this control is less fine than the rest, but this one gives you some control, that one gives you no control although it's fine. And parallelisation, this will give you some speedup. These are the properties. Questions please.

Gianpaulo Bella: This may be more a comment on the original puzzle than on your work, but this is the first time I've seen this. It strikes me that the variability in the actual solving time of the posed puzzle is going to be so wide that the time itself is not easily useable for practical purposes. I'll try to make myself clear. If you are the one setting the puzzle, then you have control over the number of multiplications that someone has to do. Now to translate this into clock time, one has to take into account the real time it takes to make the multiplications, which you assume here are the most expensive thing. Firstly it is not given that the solver's computer will be alike, so the solver might only solve it in say ten times this amount or a hundred times this amount. And secondly, Moore's law and all that stuff, this could actually shrink the time. So if we have this huge variability of several orders of magnitude from the actual time that this number of multiplications translates to, how can I build any sensible real world thing? "I want this to expire in 50 years, plus or minus three orders of magnitude," doesn't mean much.

Reply: Yes, I think your comment is valid, as long as there is such variation[4].

Ross Anderson: What's wrong with just saying, solve a puzzle that uses a block cipher such as, say, Serpent with 97 bits of the key unknown. Then you're relying only on Moore's Law and your assumptions surrounding that, you're not relying on assumptions about somebody finding the more neat parallelisable way of doing small discrete log.

[3] By induction, since –1 is a quadratic non-residue and both the non-trivial square roots of +1 have negative Jacobi symbol.

[4] It would be OK on a physically sealed box like a time capsule, that could not be dynamically re-programmed.

Gianpaulo Bella: I think this is not viable because this is something that could be easily parallelised, it doesn't have the sequential properties.

But also there's a problem with the key, the amount of time to get the secret key is only probabilistically bounded.

James Malcolm: Is the expectation here that the puzzle will *have* to be solved, or is the puzzle there to be solved if something goes wrong?

Reply: This depends on applications, here I have assumed that the puzzle has to be solved.

Matt Blaze: Since multiple messages can be encrypted under this key, the architecture that might make sense for this would be: at various intervals, using some large public process in which many people are contributing bits in some way to ensure this is secret, we all agree on what this key will be in such a way that nobody actually knows what it is. And then there is a single key for which we make some estimate when it is due to be solved, and a large and very well funded effort at finding it. Now technology advances could make it happen sooner and problems could make it happen later if the effort declined (or funding did), but it's probably easier to estimate the advance in technology at the high end than at the low end.

Ross Anderson: There's a weird thing happening here, which is we live in an almost mediary era of computational history, where the best processors are the ubiquitous ones to very close approximation. The Cray doesn't buy you very much more than a desk processor. Now this wasn't the case twenty years ago and it may not be the case twenty years hence, once there are computing devices of single electron memory or whatever, so the advantage that the agency has over graduate students may open up again.

Matt Blaze: It may open up again. But I like the notion that you have a large public process that has many secrets that it's protecting, what the information-hiding workshop described as angry mob group analysis. And I think that there's an element of this, too, if society wants this secret sooner it can devote more resources to it but it may be expensive.

Ross Anderson: As people move from quadratic sieve to number field sieve, the computation relies more on the linear sieving than on the linear relations stage, so parallelising certainly becomes inherently less powerful.

Trust Management and Network Layer Security Protocols

Matt Blaze[1], John Ioannidis[1], and Angelos D. Keromytis[2]

[1] AT&T Laboratories – Research
{mab,ji}@research.att.com
[2] Distributed Systems Labs
CIS Department, University of Pennsylvania
angelos@dsl.cis.upenn.edu

1 Introduction

Network-layer security among mutually trusting hosts is a relatively straightforward problem to solve. The standard protocol technique, employed in IPSEC [KA98], involves "encapsulating" an encrypted network-layer packet inside a standard network packet, making the encryption transparent to intermediate nodes that must process packet headers for routing, *etc.* Outgoing packets are authenticated, encrypted, and encapsulated just before being sent to the network, and incoming packets are decapsulated, verified, and decrypted immediately upon receipt[IB93]. Key management in such a protocol is similarly straightforward in the simplest case. Two hosts can use any key-agreement protocol to negotiate keys with one another, and simply use those keys as part of the encapsulating and decapsulating packet transforms.

In many applications, security at the network later has a number of advantages over security provided elsewhere in the protocol stack. Network semantics are usually hidden from applications, which therefore automatically and transparently take advantage of whatever network-layer security services their environment provides. Especially importantly, the network layer offers a remarkable flexibility not possible at higher- or lower- abstractions: security can be configured end-to-end (protecting traffic between two hosts), route-to-route (protecting traffic passing over a particular set of links), edge-to-edge (protecting traffic as it passes between "trusted" networks via an "untrusted" one), or in any other configuration in which network nodes can be identified as appropriate security endpoints.

Fortunately, the design of encapsulation techniques for basic authentication and confidentiality services is not a conceptually difficult problem, and network-layer security protocols, such as IPSEC, have matured to the point of being standardized and implemented by commercial vendors.

A harder problem, however, and one that current standards for network-layer security do not address, is the management of the policy governing the handling of packets on the way in to or out of a host running the encapsulation protocol. By itself, the security protocol protects packets from external tampering and eavesdropping, but does nothing to enforce a policy as to which hosts are

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 103–108, 2000.

authorized to exchange particular kinds of traffic. In many configurations, especially when network-layer security is used to build firewalls and virtual private networks, such polices can be quite complex.

Central to the problem of engineering policy mechanisms for network security is the tradeoff between expressiveness and performance. Unfortunately, many configurations demand a high level of both.

In this position paper, we examine the problem of managing policy in network-layer security protocols, and propose a trust-management architecture for network-layer security that may satisfy both the expressibility and the performance issues.

2 IPSEC Policy Architecture

Let us examine the architecture of network-layer security more closely, using IPSEC as a specific example. In this environment, policy must be enforced whenever packets arrive at or are about to leave a network security endpoint (which could be an end host, a gateway, a router, or a firewall).

When an incoming packet arrives from the network, the security endpoint first determines the processing it requires:

- If the packet is not protected, should it be accepted? This is essentially the "traditional" packet filtering problem, as performed, *e.g.,* by network firewalls.
- If the packet was encapsulated under the security protocol:
  - Is there correct key material (usually contained in a data structure called a "security association") required to decapsulate it?
  - Should the resulting packet (after decapsulation) be accepted? A second stage of packet filtering occurs at this point. Notice that a packet may be successfully decapsulated and still not be accepted (*e.g.,* a decapsulated packet might contain an illegal network source IP address such as 127.0.0.1).

A security endpoint makes similar decisions when an outgoing packet is ready to be sent:

- Is there a security association (SA) that should be applied to this packet? If there are several applicable SAs, which one should be selected?
- If there is no SA available, how should the packet be handled? It may be forwarded to some network interface, dropped, or queued until an SA is made available, possibly after triggering some automated key management mechanism such as the IPSEC ISAKMP protocol[HC98].

Observe that because these questions are asked on packet-by-packet basis, policy filtering must be performed, and any related security transforms applied, quickly enough to keep up with network data rates. This implies that in all but the slowest network environments there is insufficient time to process elaborate

security languages, perform public key operations, consult large tables, or resolve rule conflicts in any sophisticated manner.

Implementations of network layer security services, including IPSEC and most firewalls, therefore, usually employ very simple, filter-based languages for configuring their packet-handling policies. In general, these languages specify routing rules for handling packets that match bit patterns in packet headers, based on such parameters as incoming and outgoing addresses and ports, services, packet options, *etc.*[MJ93]

However, packet-level filtering – necessary as it might be – is not the interesting problem.

3 Policy and Security Associations

A basic parameter of the packet processing problems mentioned in the previous section is the question of whether a packet falls under the scope of some Security Association (SA). SAs contain and manage the key material required to perform network-layer security protocol transforms. How then, do SAs get created?

The obvious approach involves the use of a public-key or Needham-Schroeder [NS78] based key distribution scheme as the basis for a protocol that creates a new SA with whatever host attempts to communicate unsecured traffic in a manner that fails the packet-level security policy. At least one currently-distributed IPSEC implementation does just this, with the aim of performing "opportunistic encryption" whenever possible.

Unfortunately, protocols that merely arrange for packets to be protected under security associations do nothing to address the problem of enforcing a *policy* regarding the flow of incoming or outgoing traffic. Recall that policy control is a central motivation for the use of network-layer security protocols in the first place.

In general, and rather surprisingly, security association policy is largely an open problem – one with very important practical security implications and with the potential to provide a solid framework for analysis of network security properties.

Fortunately, the problem of policy management for security associations can be distinguished in several important ways from the problem of filtering individual packets. In particular:

- SAs tend to be rather long-lived; there is "locality of reference" insofar as hosts that have exchanged one packet are very likely to also exchange others in the near future.
- It is acceptable for SA creation to require substantially more resources than can be expended on processing every packet (*e.g.,* public key operations, several packet exchanges, policy evaluation, *etc.*)
- The "output" of negotiating an SA between two hosts can provide (among other things) parameters for lower-level packet filtering operations.

A trust-management system[BFL96], such as *KeyNote*[BFK99], may be of value here.

4 A Trust Management Architecture for Network Layer Security

The problem of controlling SAs in a network-layer security protocol is easy to formulate as a trust-management problem. Trust-management systems are characterized by:

- A method for describing "actions," which are operations with security consequences that are to be controlled by the system.
- A mechanism for identifying "principals," which are entities that can be authorized to perform actions.
- A language for specifying application "policies," which govern the actions that principals are authorized to perform.
- A language for specifying "credentials," which allow principals to delegate authorization to other principals
- A "compliance checker," which provides a service for determining how an action requested by principals should be handled, given a policy and a set of credentials.

The trust-management approach has a number of advantages over other mechanisms for specifying and controlling authorization, especially when security policy is distributed over a network or is otherwise decentralized.

In the case of SA policy, the "actions" would represent the low-level packet filtering rules required to allow two hosts to conform one another's higher-level policies.

This suggests a simple framework for trust management for Network- Layer Security:

- Each host has its own trust-management-controlled policy governing SA creation. This policy specifies the classes of packets and under what circumstances the host will initiate SA creation with other hosts, and also what types of SAs it is willing to allow other hosts to establish.
- When two hosts discover that they require an SA, they each propose to one another the "least powerful" packet-filtering rules that would enable them to accomplish their communication objective. Each host sends proposed packet filter rules, along with credentials (certificates) that support the proposal. The trust structure of these credentials is entirely implementation dependent, and might include the arbitrary web-of-trust, globally trusted third-parties, or anything in between.
- Each host queries its trust-management system to determine whether the proposed packet filters comply with local policy and, if they do, creates the SA containing the specified filters.

Other SA properties might also be subject to trust management policy. For example, the SA policy might specify acceptable cryptographic algorithms and key sizes, the lifetime of the SA, logging and accounting requirements, *etc.*).

Our architecture divides the problem of policy management into two natural components: packet filtering, based on simple rules applied to every packet, and trust management, based on negotiating and deciding which such rules are trustworthy enough to install.

This distinction makes it possible to perform the per-packet policy operations at high data rates while effectively establishing more sophisticated trust-management-based policy controls over the traffic passing through a secure endpoint. Having such controls in place makes it easier to specify security policy for a large network, and makes it especially natural to integrate automated policy distribution mechanisms.

An important practical problem in introducing security policy mechanisms is the transition from older schemes into the new one. Existing IPSEC security policies, which are based only on packet filters, quite easily fit into the trust management framework. As the trust management mechanism is introduced, filter-based policies can be mechanically translated into trust-management policies and credentials.

5 Conclusions and Status

We have developed a number of trust management systems, and have started examining the use of *KeyNote* in the engineering of network-layer security protocols. We are in the process of implementing an IPSEC architecture similar to that described above; it is our hope that the formal nature of trust management will make possible network security configurations with provable properties. One of the most relevant features of trust management to SA management is the handling of policy delegation.

Furthermore, because KeyNote is application-independent, it can be used to "tie together" different aspects of network security, beyond just IPSEC and packet filtering. For example, a more comprehensive network security policy could specify what mechanisms are acceptable for remote access to a private corporate network over the Internet; such a policy might, for example, allow the use of cleartext passwords only if traffic is protected with IPSEC or some transport-layer security protocol (*e.g.,* SSH [YKS+99]). Multi-layer policies would, of course, require embedding policy controls into either an intermediate security enforcement node (such as a firewall) or into the end applications.

Finally, if trust-management policies and credentials are built into the network security infrastructure it may be possible to use them as an "intermediate language" between the low-level application policy languages (*e.g.,* packet-filtering rules) and higher-level policy specification languages and tools. A translation tool would then be used to convert the high-level specification to the trust-management system's language (and perhaps vice-versa as well). Such a tool could make use of formal methods to verify or enforce that the generated policy has certain properties.

There are many open, and we believe, quite interesting and important problems here.

References

BFK99. M. Blaze, J. Feigenbaum, and A. Keromytis. KeyNote: Trust Management for Public-Key Infrastructures. In *Proceedings of the 1998 Cambridge Security Protocols International Workshop*, pages 59–63. Springer, LNCS vol. 1550, 1999. 105

BFL96. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proc. of the 17th Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, Los Alamitos, 1996. 105

HC98. D. Harkins and D. Carrel. The Internet Key Exchange (IKE). Request for Comments (Proposed Standard) 2409, Internet Engineering Task Force, November 1998. 104

IB93. John Ioannidis and Matt Blaze. The Architecture and Implementation of Network-Layer Security Under Unix. In *Fourth Usenix Security Symposium Proceedings*. USENIX, October 1993. 103

KA98. S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. Request for Comments (Proposed Standard) 2401, Internet Engineering Task Force, November 1998. 103

MJ93. Steven McCanne and Van Jacobson. A BSD Packet Filter: A New Architecture for User-level Packet Capture. In *Proceedings of the Annual USENIX Technical Conference*, pages 259–269, San Diego, California, January 1993. Usenix. 105

NS78. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–998, December 1978. 105

YKS+99. T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. SSH Protocol Architecture. Internet Draft, Internet Engineering Task Force, February 1999. 107

Trust Management and Network Layer Security Protocols

(Transcript of Discussion)

Matt Blaze

AT&T Laboratories

We're going to talk about some work that's matured to the point where we're doing our next generation of it, on managing trust and policy in network layer security. Last year at this workshop my graduate student Angelos Keromytis talked about a language we were designing for describing policies and credentials in distributed systems aimed particularly at practical applications – a trust management system that you might actually want to use – called KeyNote. What I'm going to talk about is a specific application of that, and a more generally interesting problem that I think may lead to a slightly different way of looking at the designs of security protocols in practice.

I should say this is joint work with a number of people, Joan Feigenbaum and John Ioannidis from AT&T labs are also involved and of course Angelos Keromytis from the University of Pennsylvania is probably doing part of this as his doctoral thesis.

Network layer security is something that we're hearing a lot about and certainly have to be concerned with as security protocol people. Network layer security obviously protects data at the network layer. It focuses on security services that let you think of the network as being itself secure. Now the usual technique for doing network layer security in packet oriented networks is datagram encapsulation, the IPSec protocol is a good example of this. The idea of datagram encapsulation says, don't do anything to change the network or the basic protocol for routing packets in the network, instead just put a protected packet inside your conventional unprotected one. On receipt you just extract the protected packet from the unprotected one and go to a recipient and then process it again as if it had just come in, after you've decrypted it and authenticated it and so on.

Now the main advantage of doing security at the network layer is you get versatility that you just don't get in possible configurations at other layers. Any node that receives packets and sends packets can become a security end-point. Now this means you can do end-to-end encryption and authentication using precisely the same protocols that you use to protect the links, or to protect a tunnel between two networks, and in fact network layer security protocols are the basic building block for virtual private networks which has become, since we first used the term about five or six years ago in this context, a marketing buzzword that people want.

Now this versatility of course creates enormous potential security problems. The fact that network layer security can be configured, to provide any arbitrary

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 109–118, 2000.

set of security services you want, protecting any arbitrary subset of the network, in any arbitrary way you choose, means that it is very easy to design a network layer security protocol in a network layer security system that provides very different security properties from what you might have originally intended, or might naively expect it provides.

Managing security in a network layer security programme is really a matter of managing something called a security association. Now those of you who are familiar with protocols like IPSec and other protocols at this layer and of this style are familiar with the term. The security association is really the basic data structure that describes how to handle the secure packets that come in. A security association coupled with a crypto key use defines what class of packet it should be applied to, and essentially amounts to establishing a channel between two end points that agree on the correct information between their security associations. And in a network layer security protocol, like IPSec, incoming packets identify the security association in a way that the recipient can make sense of, the security association identifier is negotiated as part of the key exchange. Whether or not to accept a packet that was protected under a particular security association, after you've decrypted it and authenticated it and extracted it, is a policy decision that the network layer protocol doesn't help you with. All the network layer security protocol helps you with is getting to the point where you've decrypted it, it doesn't tell you whether or not you should accept it.

Similarly there's an analogous output policy decision that you have to make in network layer security which is, if you've got a packet to send out, which security association, if any, should you use to protect it. Where do you send it, it's similar to a routing problem, if you have multiple interfaces to choose from which, if any, you send this packet that you generated and want to send. Or should you apply encryption, should you apply authentication, that's a policy decision, again, the existence of the network layer security protocol doesn't solve that policy decision for you.

So there's this policy problem in network layer security, and I'm using policy as a pseudo technical term – this is not a definition in any formal sense. A host's policy is whatever process it's using to determine whether or not he can exchange data of a particular type with other security endpoints or with other endpoints on a network or with other network interfaces. So you need some policy for governing with which other hosts you are willing to exchange data at all, and that can amount to the question of, is there a key that you can use to correctly encrypt and authenticate this particular packet, or decrypt and authenticate the packet, and what kinds of data you can exchange.

Now in particular there are many analogous problems here, to the problem of configuring network firewalls. In the problem of configuring network layer security, it's very tempting to say, what we have is this protocol that will protect data and will protect packets, so let's use that to implement a firewall. And we'll say all data has to be encrypted in order to come into this network, and we'll only set up a security association with people who are authorised to enter the

network. Unfortunately, that's a relatively ill-specified policy and it doesn't tell you how to do it, it's merely something that you might ultimately choose to do. It's very tempting to believe that network layer security solves this problem but it merely allows you to start thinking about it as a problem.

Now there's really a hierarchy of policy problems that is presented once you have a protocol like IPSec. The one that most attention has been paid to is the problem of packet layer policy, that is how should each packet be treated in a very stateless and quick way. That is, should the packet be encrypted, should it be sent out, should it be decrypted and accepted once it's been decrypted? The way we have to enforce such a policy is with something that looks very much like a filter that has the ability to examine a packet, has the ability to examine the set of security associations that may or may not exist, and make a decision on which security association to apply to it and whether an incoming packet which is protected with the security association should have actually been protected with that.

Then there's the security association policy which is much less talked about, which is the question of which of these filters should you put in to a security association once you have one. Now there's no standard way to do this, there's no standard method for saying, when two hosts want to negotiate a security association, for negotiating what their policy is as to what kind of data is going to be transmitted with that security association, does the fact that you've established a security association mean that the end point you've established it with is allowed to send any data originating at any host or claiming to originate at any host through that tunnel and will be accepted as a given merely by virtue of the fact that it authenticated properly. The policy of whether or not that's the case is one that comes up when creating security associations.

And then there's the meta policy problem of who gets to set these policies and who gets to change them. Ultimately policy becomes less tactical and more organisational, it may be informal, ill-specified and based on a notion simply that any data connected to the network needs to be secured in some way and then that has to be implemented with whatever technical means is available.

Let me make two other observations here. First is that packet level policy, what we focused on, of what treatment do you apply to each packet associated with the security association, must be fast enough to keep up with network traffic. You have to do it on every incoming and outgoing datagram and if you have multiple security associations then you have to check on outgoing traffic, you might have to check all of these possible tactical filters to be able to this very quickly. So a filter that enforces packet policy, is by its nature limited to doing operations that can be done quickly without very much state associated with it, it's probably examining only the header of the packet and some table saying what kinds of header information are permissible, at the current time, given the end points and the particular security association that you might want to use to encrypt it. On the other hand the policy for security association creation, which filters you include, can be a bit slower, you only have to enforce it at the time that you're creating the security association. What that means is that

this can be considerably more complex, and needs only to be no slower than the key exchange that you're doing when you're creating a security association in the first place, which gives you a lot more breathing room than at every packet transfer.

Now it may also be based on a changing policy, it might be based on credentials, you might present a certificate that allows you to log into a company's network remotely. This starts to look like an abstraction that we think is useful here, that we call trust management, the problem of deciding, based on policies, credentials and relationships, whether or not to perform a particular action. So the work that we're starting to do is the question of how do you do trust management for networks.

Trust management, in spite of what its name implies, has nothing to do with finances, has nothing to do with managing trusts and it has nothing to do with how much faith you might put in your boss, which is what some of the AT&T management thought. "Oh, trust management – good idea" they said, "more people should trust management", but in fact that really runs contrary to our sensibilities. It's for answering questions of the form: should I, being a host that's performing some operation, perform this potentially dangerous action that someone out there is asking me to do. This abstraction is useful because it gives you a systematic way of identifying and managing security policies, credentials and relationships as a trust management problem, we're adding this layer of abstraction called the trust management layer that gives you these answers to this question: I want to do this dangerous thing, should I, based on some request that's coming in over the network.

Our notion of this abstraction is that the trust management system provides a compliance checking service to applications, that evaluates actions against policies based on credentials and based on who is requesting. So, slightly more formally, a trust management system is essentially a language for describing actions, which are any operations with security consequences; a naming scheme for principals, being the entities that can request actions or enforce policies; a language for describing security policies, policies governing the actions that principals are allowed to perform; a language for credentials, being some mechanism for allowing principals to delegate their authorisation to other principals; and a compliance checker that gives you the model of computation for taking actions, policies, credentials and principals, and determining whether or not these comply with the underlying policy.

So we have a trust management system that we think may be useful for this (maybe not, but the abstraction is the interesting part here), called KeyNote, that Angelos described last year. Actions in KeyNote are name-value pairs, the semantics of which are application dependent, so any application that wants to use KeyNote first decides what all of the names for actions might be and what their attributes are, and what semantics different attributes have.

Principals can be arbitrary names but most importantly can be public keys so the name of a principal can simply be its public key. This avoids the hardest problem in distributed computing, which is naming – it has always been the

hardest problem and probably will remain the hardest problem. There's a common language for policies and credentials – credentials are just signed policies – which gives you a very natural way of distributing policies in organisations, simply sign it and send it off on the network and people who want this policy applied to their data can simply send a copy of it, harmless because of the monotonicity properties the system has. And we believe it's flexible enough to represent a fairly wide range of policies, such that you could use this as the primary policy engine of many applications. What you get back is called a policy compliance value which might be *true* or *false*, that simply says whether or not the action and policies and credentials submitted with the request are in fact compliant and are consistent.

Now this suggests a simple architecture for managing trust in IPSec. In our architecture an IPSec security association contains a simple packet filter. Now this is in fact the same architecture that we used when John Ioannidis and I built the first IP network layer security protocol back in late '92. We essentially took the same engine, took the packet filter out, and are using that with IPSec. So there's a pattern matching engine that examines incoming and outgoing packet headers, in a way that's fast enough to be faster than whatever encryption you might have to do or decryption. Security association creation, when you negotiate a key, is controlled by a policy written in KeyNote. Now the action is simply a representation of the filter that you would install with the security association, so when two hosts want to create a security association, they each send to the other a description of the filter they want installed with this security association. That gets converted into a KeyNote action, KeyNote then evaluates whether or not that filter is appropriate given the key of the end point that wants to negotiate.

You can also have credentials here, so you can have distributed policy simply by having KeyNote credentials that serve as the certificate of such a system. This turns out to be fast enough to evaluate as part of the key exchange without slowing things down significantly. If you're doing a Diffie-Hellman operation in the protocol, you don't notice the trust management aspects of it.

So to give you a quick example of what this looks like. Once you have this structure you could have a policy that says, for example, anybody who has this DSA key (it's a little shorter than the one you want to use in practice, but short enough to get it to fit on the slide), for the network application where TCP and UDP packets (and it's two different transports under IP), from the source address `192.11.*.*` with the destination address `135.205.89.*`, and this is essentially a policy that says this key is allowed to tunnel packets from this network.

Question: So you only have IP headers, you don't have transport headers?

Reply: We have transport headers as well, this is in the representation of the filter. Essentially all of the fields in the IP header and in the transport header are available to be matched against in the filter. It runs faster if you go less deep into the packet.

Now the delegation mechanism would allow you to break this up a little bit, you could have a policy that says, there are some authorities that are allowed to

issue the credentials that allow logging-in to the network application for TCP and UDP, but you have an actual credential that allows a particular key to actually login in a particular instance, so this is very natural to do in the KeyNote language. But that's not what this talk is about, to look at the particulars of the language itself, we think that the architecture is the interesting aspect of this and may lead to interesting ways of building scaleable systems. The main benefit of an architecture along these lines is the extensibility aspect of it. Policy doesn't need to be hard coded and it's clearly separated out from the code that's actually implementing it. The filters have become human readable, or at least somewhat human readable, rather than embedded in C code somewhere, compiled into the kernel.

It's expressible enough to represent the policies for virtual private networks, enforce secure DNS (that is enforce the fact that you have come from the network that you claim to be coming from by name), remote access to networks, ... You can represent access control lists in this framework relatively straightforwardly. There are some interesting things you can do if you have the same policy language for, say a network layer protocol as you do for the applications and for other layers of abstraction. You might be able to rely on the existence of certain network layer security protocols and directly interpret the fact that, I know my network is protecting this aspect of it, therefore I don't need to worry about that at the application level. On the other hand you might say, ah I see my network is not protecting this therefore I must do this again efficiently. It becomes at least straightforward, or more straightforward, given a framework like this to do that.

So what are the open problems? Well for a first approximation all of them, we've very much just started this, we built simple prototypes but we haven't really figured out exactly what this is useful for. Certainly things that we think are interesting to study are policies that are based on provable properties – build a language for describing policies, you might be able to simply use the output of a system that has some provable properties that you like, and then implement them directly, if you know the formal properties in your own language.

If we can optimise credentials and policies we might be able to do this in a way that allows packet level policies to be specified in the same way, that is why I must form a trust management decision on every packet, so there's an optimisation question that comes in. You might be able to synthesise code, for example, given KeyNote credentials, that actually implements the packet filter directly. The open problem of policy and credential distribution, the key distribution and certificate distribution problem, does not go away but it changes a little bit here, the problem I mentioned of interaction, in security layers and so on. Policy discovery and negotiation is a very interesting problem that, I think, an architecture like this at least gives you a handle on being able to study it in some reasonably standardised way.

There's an Internet informational RFC, that describes the KeyNote language that you get off of my FTP site, `ftp.research.att.com/dist/mab/knrfc.txt` and a mailing list. We have a free implementation of KeyNote that you can play

with if you want, the network layer security stuff we haven't reached a point that we're giving away code, but we're certainly open to collaboration particularly if you have the benefit of not falling under US export laws. Anyway I think this is an interesting framework for looking at some very interesting problems of layers.

Ross Anderson: I have a mild concern about the security association appearing in the clear on packets, because this allows a selective denial of service attack.

Reply: Ultimately that's a hard problem. It's the security association *identifier* that appears in the packet, that's OK, but it doesn't change. The security association between two hosts has to be clear enough that a host has the ability to interpret what it is. This is telling you what encryption he used to decrypt it, so there maybe ...

Ross Anderson: Well there's a couple of ways forward, you might say that you can layer this thing on top of itself, so you provide a virtual cable between Microsoft and Compaq and the outside world doesn't see happens to the strands inside that.

Reply: Yes, you can certainly configure IPSec in that way.

Ross Anderson: Or you could say on the other hand, well the security association will just be the end-point networks which is good in any case to ...

Reply: Well OK, if you have only one security association between two end points, that's essentially what you've got. The security association is just an identifier, it doesn't actually tell you what the key is, it doesn't actually tell you what the filter is, it just gives you an index that the recipient can use to look up that data. Yes, but that's in the IPSec protocol which I disclaimed any responsibility for unless it's convenient.

Michael Roe: This is more of an architectural point about IPSec. Your policies are limited to ones that get expressed in terms of filtering all the packets without reference to any other state.

Reply: The packet level policies.

Michael Roe: Even the security association policy. I mean for example, if you have an RPC-based service that dynamically gets itself a port number and then registers itself, your high level policy is that your RPC application should be encrypted with this key, but if it keeps coming up on different ports each time the machine is booted ...

Reply: There are a couple of things I didn't tell you about the architecture. One of the parameters that's thrown into the action description is essentially the complete set of environment variables in the kernel of the machine that it's running on, it has access to any of that data. You could use that to build stateful versions of this that look at other data besides simply what packet filter would do. You could, for example, have policies that allow a particular packet filter only if certain state in the kernel has already been set, such as registration with the portmapper.

Michael Roe: It's more of a question of what is the primary route for getting information into the access control decision. The related thing is, a lot of places have dynamically allocated addresses now, and IPSec generally assumes

statically allocated addresses Again it's a case where you almost feel there should be architecturally another layer of indirection in there somewhere that lets you cope with dynamic changes.

Reply: Yes, in fact you can see the handle here, when you get IP address you just get a credential that says you are entitled to use it, it's valid for a period of time, you can have it. But these are details of how we've chosen to do things because what's interesting is the architecture.

Stewart Lee: Following on from what Mike just said, what you're doing I think is confidentiality and integrity in the information, but you don't resist traffic analysis.

Reply: Yes, that's right, except to the extent that you can do that with an IPSec tunnel.

Stewart Lee: If that resists traffic analysis.

Reply: You can still see the volume of traffic.

Stewart Lee: The volume and the traffic.

Reply: Yes, that's right, and this is a difficult problem with network layer security, because if you use network layer security to simulate link encryption you lose a lot of the nice properties of the network.

Stewart Lee: Yes, well, we've all been through this.

Virgil Gligor: Last year's topic was one of trust and we were all struggling to define what we meant by trust. You sort of illustrated a few aspects of trust and the trust management, but you haven't really provided the definition of what you mean by trust. Would you attempt to do so or would you leave it as examples?

Reply: No, we definitely choose not to try to ...

The problem is when you start getting into words like trust is that what I end up coming back to is *cares*, we've got a term "trust management" and we're using that to mean the specific way of managing policies and so on, that's not necessarily all encompassing of all of the kinds of trust that you'd want to manage.

Stewart Lee: Also it pleases your management.

Reply: Yes, that's right, it seems to please our management that we're going out there and telling people that they should be trusted.

Bruce Christianson: One of the nice things about the policymaker approach is that in contrast to a lot of other approaches it doesn't commit itself to the particular semantics of trust, I think in the main it provides a set of mechanisms that are actually very neutral.

Reply: Well, in fairness I'm not sure how neutral they are. There are some meta-assumptions that we've made and I haven't even tried to identify what all of them are. I think that may be your point, we're assuming a framework here.

Bruce Christianson: No it's not completely neutral but there are a large number of different semantic communities that will agree about the bit processing that needs to be done.

Stewart Lee: Are these meta-assumptions documented in the web page?

Reply: No, I'm not sure what they are.

Bruce Christianson: You made a very important point early on, you said that of course network security policies don't solve problems with the other layers but they're a very good way of flushing those problems out into the open where you can see them and start thinking about them.

Reply: Yes, I think network layer security is extremely useful for it's flexibility. But on the other hand, part of the problem with the Internet style of protocol stack is that it doesn't provide good communication between layers. For example you can't, in any standard way, have an application find out the security properties of the network. When you've created a socket there's no standard way of saying, alright tell me what the security properties of the socket are, or tell me what security properties I require of the socket, and it certainly would be nice if there were, but there aren't.

Bruce Christianson: Well the filtering is sort of a backdoor way of getting some information about what was in the header.

Reply: Well, but that information doesn't get passed above or below, it stays at the network layer's surface.

Bruce Christianson: Oh sure, but the fact that you got the packet means that it's passed the test.

Reply: Yes, that's right. But one of the open problems is, that if you have an application like a web browser, for example, you might say well, if I know that the connection to this service that I've got is protected under IPSec, then I know I don't need to further protect it using SSL encryption because it's already being protected at the network layer, assuming that the network layer is providing security adequate for the threat that you're using SSL for in the first place. On the other hand there's no easy way to find out in the browser itself. The network layer can have a policy that says, when packet not encrypted must do IPSec, so you think you can go downward in that way, but can you go back up, and in a standard manner? So interfaces for doing that remain an open problem and understanding what the implications are of that kind of layer collapsing is certainly an interesting question.

Virgil Gligor: How much state do you have to keep to enforce your network layer security policies?

Reply: I don't know yet actually, that's an interesting question. We are just starting this so what we haven't done is try to build any unusual or complicated policies, what we're tried to do at this point is only replicate existing policies.

Virgil Gligor: Here is why I'm asking the question. Presumably I receive a packet, and the packet has some information about identities, structures, which I have to check. To rely on that information from the packet I have to do perhaps some cryptographic checking which means all of a sudden that I am relying on an area called cryptography, symmetric key possibly, which relies on another area called key distribution and possibly public key cryptosystems, which relies on PKIs, ... So have you looked at this dependency graph that is required by your trust management, and how much could you collapse this dependency graph?

Reply: Yes, I think that's a very good question, it's not one I've looked at, but I agree that it's a very interesting question.

Virgil Gligor: That's where you are going to hit upon one aspect of trust which is that of depending on somebody to provide some services for you which you cannot check or you cannot provide for yourself.

Reply: Yes, this leads to an unrelated point but you've reminded me of it. I think that the single greatest threat to practical security is our tendency as people like clean designs, to provide nicely paramaterisable protocols and services, which merely pushes all the hard questions off to the person who has to actually configure it, and can configure it in bad ways. A much better service to users of security protocols would be to design the protocols with parameters that are simply secure. Allowing people to choose between a 65 bit key and a 73 bit key, I think does not serve the interests of security very well.

Virgil Gligor: That is one of the questions of trust, the other one is the one that I just mentioned, it's reliance on other mechanisms and services outside of the layer that you are taking about.

Reply: That's a central and very interesting question in our community in general.

Bruce Christianson: But you don't do any cryptographic processing on the packet headers, you take them at face value when you do the filtering.

Reply: In IPSec there's an outer packet and an inner packet, you take the outer packet at face value, that tells you how to process the inner packet, if it's been corrupted then whatever it tells you won't work. On the inner packet you then apply filtering that says, well now that I've extracted this header information, do I actually trust that security association with this packet, and you do the analogous thing on outgoing.

Virgil Gligor: Presumably, though, the outer packet is just hints.

Reply: The outer packet is just hints, and provides the information required for the network to get it from one place to the other.

Stewart Lee: But in fact the outer packet has an even further outer packet which has to do with addressing.

Reply: Yes, that's right, a link layer.

I think this is an interesting way of looking at the problem because it at least exposes what the problems are.

Issues in Multicast Security

Francesco Bergadano[2], Davide Cavagnino[2], and Bruno Crispo[1,2]

[1] Computer Laboratory, Cambridge University, England
[2] Dipartimento di Informatica, Università di Torino, Italy

Abstract. Multicast communications introduce significant novelties both at the Network and Transport layers. The relevant applications, on the other hand, have special requirements, determined by the real-time and multimedia nature of the data and by the large number of receivers. This paper first discusses the basics of multicast routing and transport, and the concept of a multicast "session". Then, it analyses the security services that are needed and the difficulties that arise in this context. The techniques that have been proposed for solving different multicast security problems are briefly surveyed.

1 Introduction

Multicast communications have generated increasing interest during the past few years. This is due, in principle, to the obvious potential for improvement in the use of network resources: data is sent to a number of different recipients in just one transmission, instead of requiring the setup and the maintenance of distinct unicast connections. This means substantial savings both in terms of network load and in terms of server operation. However, multicast communications are generally more complicated and cause a number of difficult problems at different levels. In particular, the lack of multicast support in most Internet routers has made related technologies largely unavailable. Today, the widespread use of multicast-enabled applications and networks is possible and even likely. The reasons for this are twofold.

On the one hand, the experience of the MBONE network (see, e.g., [32,19], and the references contained in www-mice.cs.ucl.ac.uk/multimedia, www.ipmulticast.com) has pushed the technology by demonstrating that, although many problems remain to be solved, the technology is mature and sufficient for supporting high quality applications. The MBONE is overlayed to the standard Internet by tunneling multicast communications via IP unicast, and having routers at each end of the tunnel support multicast routing. Although this was initially done with normal workstations, the main commercial routers are now supporting this function.

On the other hand, the technology is being pulled by changes in the size of the Internet and in the way people use it. Multicast conferencing, allowing to hold meetings and interactive conferences over the network, may witness increased use in the near future [22]. But also the more traditional and more widely available WWW medium is changing and moving towards a kind of usage that would be

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 119–131, 2000.

more appropriate for multicast transmission, at least in some cases. Many sites offer streaming audio and video, that may represent significant amounts of data being transmitted to one user, and then transmitted again and again for the requests that follow. Recently, the concept of 'push' technology has been proposed and commercialized. 'Push' applications lie between the old-fashioned 'passive' receiving of broadcast information, and the type of on-demand interaction of the WWW. 'Push' communications are best implemented with multicast, especially when audiences are large. This may change the way the network is used today, where some Web sites report receiving millions of requests each day: this means tremendous duplication of data being transmitted over the same physical media, and sometimes leads to the denial of service that follows excessive load at the server.

A multicast session, or 'conference', takes place as follows [14]:

- an initiator creates the session by obtaining an IP multicast address, and by defining its characteristics, including the media types to be transmitted, the scope and the duration;
- the session is announced [18,24,20], or users are invited to take part in the session [21]
- applications for receivers interested in the session 'join' the multicast group and inform their local routers [8,13,6]
- senders send 'to the group' (send to the IP multicast address)
- receivers may reserve an appropriate quality of service (QoS) [29,3];
- routers work in such a way that information is delivered to all receivers in a most efficient way and so that QoS reservations are met.

Typically, a multicast session includes audio and video streams that have real-time requirements. A transport protocol has been defined [34,33] that attaches timestamps to packets, so that information that is not timely may be discarded, and streams may be played back with correct sequencing. Normally, real-time data that arrives late or does not arrive is not retransmitted, although forward error correction may be implemented. Data that requires reliability (e.g. multicasting of text, software or files), requires different transport protocols, allowing for retransmission, and much effort is being devoted to this issue [27,35]. In summary, multicast users should be able to receive (and possibly, send) data of different kinds, some requiring sufficient QoS and timeliness, others demanding the completeness and correctness of the incoming information.

Is security an issue in this context? In this paper, we argue that it is. However, we also show that the techniques available for unicast security are not easily ported to multicast applications. In particular, the absence of a two-party connection makes it difficult to use standard security protocols above the transport layer. Moreover, the real-time, streaming nature of the information being transmitted creates the need for security services that are novel, while usual concepts such as integrity and privacy need to be reinterpreted. In general, all practical security solutions must be tightly coupled to the multicast protocol suites that are in use, and must be scalable to large numbers of communicating parties. We

will then first discuss the state of the art in protocols for multicast networks, only then will we address a number of security issues.

2 Multicast Protocols

Multicasting is possible if it is supported at the network layer, so we first discuss the associated routing and group joining protocols. ¿From the point of view or applications, the concept of a multicast *session* is fundamental, just as the concept of a *connection* in unicast. We will then address session creation and management. Finally, content must be received in a way that is suitable for the application requirements. This may mean either timeliness or reliability, and we discuss the associated transport protocols.

2.1 Multicast Groups

In multicasting a source sends datagrams to a class D IP address: class D addresses (from 224.0.0.0 to 239.255.255.255) are bound to *multicast groups*, each address specifying a group. A receiver interested in receiving data destined to a group has to subscribe to that group. Thus, receivers need to use a protocol to join a particular group: this protocol is the Internet Group Management Protocol (IGMP). A sender may or may not belong to the group to which it is sending. In the discussion throughout this paper two important points should be kept in mind: the first one is that any host may transmit to a particular multicast group; the second is that any host may receive the packets a sender is transmitting, the sender being generally unaware of the receivers.

IGMPv1 [8] is used for communications between hosts and multicast routers on the same subnet. With this protocol, a host informs its router that it is interested in receiving data addressed to a particular group. A multicast router on the subnet periodically multicasts membership query messages on the subnet and a host responds with a membership report, one for each group it belongs to: hosts interested in the same group hearing a report from another member cancel their own report (packets addressed to a group will be sent on the subnet if *at least* one member is present on the subnet); queries and reports are addressed to the all-hosts group (224.0.0.1). IGMPv2 [13] adds more functionality to IGMPv1: it has support for fast leave of a host from a group (thus reducing traffic on a subnet), and the election of the querier router (on each subnet there must be one and only one designated router that issues membership query messages). IGMPv3 [6] defines support for sender filtering.

2.2 Multicast Routing

Multicast routers need protocols to exchange host membership information (learned using IGMP) in order to deliver data from sources to receivers interested in a group. The most widely used routing protocols are the Distance Vector Multicast Routing Protocol (DVMRP [36,28]) and the Protocol Independent Multicast (PIM, with two variants, Sparse Mode [12,10], and Dense Mode [11]).

DVMRP combines RIP and the Truncated Reverse Path Broadcasting TRPB algorithm [9]; it performs only multicast routing, and is implemented in the program *Mrouted*. The routing information between routers is exchanged by means of IGMP messages. Using TRPB the routers build a routing tree for each multicast group, and each router discovers its position in this tree in terms of the interfaces to which a received multicast packet should be sent to. *Mrouted* accepts the definition of tunnels in addition to physical interfaces on which the kernel may receive and relay multicast packets: a tunnel is a virtual link between two routers running *Mrouted* that allows to send multicast datagrams encapsulated in unicast datagrams to connect multicast islands within a unicast network. This led to the construction of the Multicast BackBone, MBone [32], overlayed on the unicast Internet. Note that the routing of the multicast packets is performed by a multicast kernel in the operating system, and that the routing protocols like PIM and DVMRP only build the routing tables. DVMRP uses *prune* and *graft* messages between *mrouted* to refuse or request packets addressed to a particular group.

PIM has two variants, SM and DM, both of which require an underlying unicast routing protocol (but none in particular) that maintain routing table information; this information is used to build the multicast distribution trees. PIM-SM is more efficient when the receiving hosts are distributed sparsely across the Internet, whereas PIM-DM is more efficient when the receiving hosts are distributed in a dense and uniform manner. PIM-DM is similar to DVMRP, but as opposed to DVMRP requires a unicast routing protocol for the reason previously outlined. If the multicast group is sparsely distributed, it is inefficient to flood the network (as DVMRP does) with multicast packets because there will be a high probability of triggering prunes; instead it is more efficient to wait for explicit joins before sending packets: in this case PIM-SM is more efficient. PIM-SM requires a router working as Rendezvous point for each multicast group: a host sending to a group has to send to the Rendezvous point for that group, and receivers subscribe to the group sending join messages for that group to their routers that will forward these joins towards the Rendezvous point. This procedure creates a shared distribution tree. A router having attached receivers on its subnet may switch to the so called *shortest path tree* directly joining the sender bypassing the Rendezvous point and, after that (to avoid losing packets) sending prunes towards the Rendezvous point.

2.3 Session Management

A session is defined by a set of media (audio, video, text, graphics etc.) being transmitted for a period of time to a set of addresses of the form address/port. To distribute the information related to a session, an announcement should be sent to, at least, the people that may be interested in the session and are authorized to join it. Given that a session is announced using the packets of the underlying network, it is possible to limit the hops made by the packets (i.e. the scope) inserting a small Time-To-Live in the packet; the TTL is a number that in unicast packets specifies how many hops a packet may go through. The

same happens for multicast packets, with the difference that, when a multicast packet is encapsulated in a unicast packet to go through a tunnel, two TTLs are involved, one for the unicast path in the tunnel, the other for the multicast path where the tunnel counts for one multicast hop (a packet is relayed on an interface or a tunnel only if its TTL is at least equal to the threshold associated with the multicast interface or the tunnel). Another way to limit the scope reached by a packet is by administrative scope: a set of multicast routers may be configured to make a boundary for packets addressed to a set of multicast addresses; in this way multicast packets may be blocked from traversing a boundary.

A session may be created in two ways: by means of an announcement or by sending an invitation. A session is advertised by announcing it on a particular well known multicast address (and port) with the same scope (either determined with the TTL or administratively) as the session it is announcing: in this way the announcement reaches all the destinations that will be able to receive the multimedia data of the session. The Session Announcement Protocol (SAP [18,24]) describes the characteristics of a session announcement, for example, its scope and the regular interval period between announcements. Using SAP it is even possible to delete and modify sessions. This creates the problem that only authorized people should be able to perform these actions. Deletion of a session should in any case occur when no announcement is heard for a certain period of time on the receiving side. A widely used tool for announcing sessions (and receiving announcements) on the MBone is *sdr* (Session Directory), an evolution of Van Jacobson's *sd*; this is a multi-platform tool and plays the role of a television guide for the MBone. One multicast address (and UDP port) is used for announcing TTL-scoped announcements (224.2.127.254, port 9875), and another address is used for an administrative scope (the address is administratively chosen, but *sdr* announces on 239.255.255.255). Thus, it is sufficient for *sdr* to join and listen to particular addresses to hear about announced sessions. Announcements may be authenticated by means of an authentication header built with an asymmetric algorithm, and may be encrypted using any encryption algorithm, even if the use of DES in CBC mode is suggested.

The meaningful information about multimedia sessions announced by SAP is formatted according to a specific protocol, the Session Description Protocol (SDP [20]). The information contained in an SDP payload describes the session in terms of type of media (audio, video, text etc.), the transport protocol (e.g. RTP or UDP, see below), the coding format of the media (e.g. PCM, GSM, H.261, H.263 etc.), the multicast (or unicast) address and port on which a particular media is being, or will be, transmitted. Moreover, SDP can transmit information about the time the session will be held and how long it will last, the URL related to this session and contact information. All this information is encoded in a sequence of ASCII strings defined by a simple grammar. Using the information delivered by SDP, the receiving host knows how to join a session in terms of programs used to decode the data and their initialization parameters (including the multicast address and port).

2.4 Multicast Transports for Real Time Data and for Reliability

The normal way multicast packets are sent over the Internet is through UDP/IP. For applications having real time requirements, the Real-time Transport Protocol (RTP, [34]) has been defined (RTP usually runs over UDP). The services offered by RTP are the identification of the payload type (i.e. the media transmission format), the timestamping and sequence numbering of the data packets and the monitoring of the transmission. All this information is inserted in the RTP packet header. Each source of real time data is distinguished by means of an identifier carried in each packet sent from that source. The timestamp in each packet indicates the sampling instant of the first octet in the RTP data packet (this timestamp may be derived from a local clock): this allows a correct reconstruction of the timing at the receiver (packets may arrive with jitter and not in order, so a playout buffer is needed). The timestamp refers to the particular real time flow, but may not be used for synchronizing two or more flows (e.g., to synchronize audio and video): for this functionality the Real-time Transport Control Protocol, RTCP, which is associated to RTP is defined; its packets provide the relationship among the real time and the timestamp-clock of a sender in order to allow for the synchronization of different flows belonging to the same sender. There are various types of RTCP packets, which are used to send information about the reception quality, and data about the senders and receivers, like personal names, telephone numbers, e-mail addresses etc. RTP also defines the concepts of translator (that, for example, translates a media encoding from one format to another) and mixer (that combines RTP packets from different sources making time adjustments to synchronize the flows).

Some multicast applications may have reliability requirements, thus various reliable multicast transport protocols have been proposed. Building a general reliable multicast transport protocol is not trivial since different applications may require different degrees of reliability, and even the structure of the application and its security requirements may force the use of particular protocols (e.g., only the sender of a packet may retransmit it). Here we discuss two of them, namely the Pragmatic General Multicast (PGM [35]) and the Reliable Multicast Transport Protocol (RMTP [27]). Other transport protocols that provide for reliability with different frameworks and techniques are the Scalable Reliable Multicast (SRM [15]) and the Reliable Multicast data Distribution Protocol (RMDP [30]).

PGM is a multicast transport protocol for the reliable ordered transmission of data (without duplication) from N senders to M receivers. PGM may be extended to support applications such as news updates and real time audio and video (for example adding reception quality reports and Forward Error Correction FEC). In PGM a source multicasts data packets (within a sliding window which is advanced in the absence of NAKs) and receivers unicast NAKs for the packets they detect as missing from the sequence. PGM defines the network element entity that forwards NAKs to the source and confirms them downstream with NAK Confirmations (NCFs). Another function of the network elements is to forward data packets along the distribution tree. Retransmission of data

may be performed by the source, or by a Designated Local Receiver in response to a NAK, or by a receiver in response to an NCF. As defined, the protocol prevents NAK implosion. The source sends at intervals Source Path Messages (SPM) that specify the structure of the distribution tree from the source towards the receivers. This information is used by the network elements to learn their position in the tree. SPM messages also announce that an advancement of the transmission window will be performed by the sender.

RMTP runs over UDP and its functionalities are the delivery of ordered data, without losses, from a sender to a set of receivers. The protocol groups the receivers into regions, assigning to each region a Designated Receiver, DR. Then, a tree hierarchy (with k levels) may be built connecting the various regions in the tree. Data is ACKnowledged from receivers to the local DR; each DR then acknowledges its upper DR in the hierarchy and so on towards the sender. When a receiver detects a data loss, it requests a retransmission to its local DR: this mode of operation results in reliability, low end-to-end delay, no ACK implosion. Retransmission of a packet may be performed using multicast or unicast depending on the number of receivers requesting the lost data.

3 Security Issues

As the use and the standardization of multicasting makes progress, there is a corresponding increase of interest in security services that are needed and yet suitable for implementation in this new context. A working group within IETF dedicated to multicast security has been formed (www.ipmulticast.com/community/smug), and general papers concerning multicast security can be found, e.g. [7,23]. In general, there are many and important motivations for multicast security, leading to novel or modified security concepts and requirements. However, two observations are important. First, the security services that are needed may be novel, and the usual concepts of privacy, integrity, and access control must be interpreted with reference to a profoundly modified context. Second, the available security technology is generally insufficient or impractical. We consider a number of distinct security objectives and briefly survey some of the work done in each area.

3.1 Group Data Privacy

Although multicast and broadcast communication is by definition not totally private, one may wish that cleartext data be visible only by the members of the multicast group. Privacy may be required for conferencing, but also for protecting content that is broadcast to paying subscribers. This is a more difficult issue than it is in unicast transmissions, for a number of different reasons. First, individual receivers are not explicitly known at the network level: receivers join the group by sending an IGMP message to their local router, and this action is not directly visible to the senders. Second, a group with a large number of receivers

will cause difficulties for key distribution and revocation. Key distribution in multicast applications has been studied extensively (see, e.g., [7,4,26]).

The simplest solution to key management is having a key distribution center (KDC). The KDC generates symmetric session keys and unicasts such keys to group members, encrypted with shared long term keys or with asymmetric techniques. This solution is not scalable and has a single point of failure, therefore it does not seem to be suitable for multicast applications. A somewhat improved scheme, based on the same idea, has been standardized as the Group Key Management Protocol in RFCs 2093 and 2094. In this standard, a per-group KDC is introduced, and is called a Group Controller. Again, large groups may be problematic.

To overcome, in part, these problems, the Scalable Multicast Key Distribution standard (RFC 1949), delegates the distribution of keys to routers, while the Group Controller is still responsible for generating keys and passing them to the higher level routers in the multicast distribution tree. This solution is attractive, but routers must be trusted, and reference to a static multicast routing protocol is made. It is not clear how the solution may be generalized to dynamic routing.

Yet another interesting key distribution scheme is introduced in [26]. Here, a Group Controller is used, but it is complemented by so called Group Intermediaries (GIs). Each GI is responsible for a subset of the group members, and uses a different symmetric group key for those group members. The GIs also share a symmetric key with the Group Controller. When receiving multicast information from the Group Controller, the GIs decrypt it and re-encrypt it with their subgroup keys. This is helpful when a group member leaves: only the subgroup of that member need change its key, under the control of the corresponding GI. However, the scheme requires multiple encryption, that may cause delays.

Instead of distributing keys from some kind of a Group Controller to individual members, one may chose to have all members collaborating so as to obtain a common key. Generalizations of the Diffie-Hellman key agreement scheme to N parties have been proposed (see, e.g., [2]), that may be useful in the managing of a secure multicast group. Such protocols consider the possibility of members leaving or entering the group, with the corresponding necessity for the remaining members to reconstruct a key. The new key should be independent from previously used keys, and yet the new key agreement (Auxiliary Key Agreement) should be more efficient than the one performed at the beginning (Initial Key Agreement).

3.2 Protection of Multicast Content

Even when data is correctly encrypted and made available only to authorized group members, high quality multimedia content must be protected against illegal reproduction by the authorized receivers. However, standard watermarking techniques are not usable, since the same content is transmitted to all.

In [1] the authors propose an encryption scheme that has the unusual property of changing only few output bits when few bits in the keys are changed.

This is the opposite the well-known avalanche effect of most encryption algorithms. By giving authorized users slightly different keys, the obtained cleartext bears a different watermark for each user, so that redistributors can be identified. Yet, the encrypted content is the same for all and thus it can be distributed in broadcast or multicast networks. However, colluding receivers may eliminate the watermarks by combining their keys.

A different technique is proposed in [5], under the name of "watercasting". There, different contents are sent to individual receivers so that unauthorized reproduction may be traced. One simple way of doing this is by unicasting content to receivers, but this represents a suboptimal use of network resources. Instead, the watercasting technique suggests to send content with some degree of packet duplication, similarly to what is done for forward error correction. However, duplications contain differences so as to achieve watermarking. Routers eliminate different duplications of the same piece of information along different branches of the multicast distribution tree. As a consequence, receivers obtain marked copies of the content, and are discouraged from reproducing it. This is more efficient than unicasting different versions because the number of duplications is proportional to the depth of the tree, while the number of receivers may be as large as the number of leaves.

Yet another approach [17] proposes to multicast corrupted content, and unicast correction codes, so that the corrected plaintext bears a watermark. Since the correction codes are much smaller than the content being multicast, the technique is adequate for large numbers of receivers.

3.3 Individual and Group Authentication

Data may have to be authenticated as coming from the group, and not from unauthorized outsiders. More meaningfully, data may have to be authenticated with respect to the individual sender within the group. Individual authentication may be implemented with digital signatures, but the computational cost is high, and this is especially a problem in multicast applications, that already load client computers with multimedia processing. More efficient signature schemes are available and may alleviate these problems. In particular, online-offline signatures [31] allow the signer to perform some of the work "offline", before the text to be signed is actually available, and this may be useful in some implementations of multicast multimedia transmissions. In [16] a solution is proposed, where one-time signatures are used, that are significantly more efficient than traditional digital signatures. However, the length of the authentication information is increased.

If implemented with symmetric keys, other inefficiencies arise, including key distribution and excessive overhead on network resources. In fact, even after a symmetric key is shared between every pair of group members, we will have to append N-1 MACs to each multicast message, where N is the size of the group. This is unacceptable in many practical situations, especially for large groups. A more efficient solution, still based on symmetric MACs is outlined in [7]. In this scheme, the number of MACs does not depend directly on the

number of group members. However, colluding group members may attack the authentication scheme and produce text that appears to be authenticated by another individual group member.

Authentication may be less important for audio and video transmissions, and essential for the kind of data that also requires reliability, e.g. text, software, files. If conferencing over the Internet becomes widespread, conferences that have some associated value will typically demand authentication for some of the data streams being exchanged. Authentication may also be important for some kind of broadcast channels, such as stock quotes, business surveys and news.

3.4 Accounting/Billing and Theft of Service

Currently, multicast is served with a best effort strategy, like all other Internet traffic. In the near future, some traffic will be prioritized with simple schemes, mostly suitable for two-party voice communications. Later, conferencing and multicasting/broadcasting with suitable QoS will need more complex measures, including some form of reservation and the possibility of some form of accounting or billing of the traffic. The RSVP [29,3] protocol has already standardized the reservation messages, to be sent by receivers to the network (to the routers). Security of reservations and session management has been considered [3,24], but if the individual traffic is not authenticated, theft of reserved traffic is possible and easy, and auditing/accounting/billing is impossible.

Individual authentication is not only needed to guarantee to other parties the authenticity of the source of the information sent through multicast channels, but probably most important it is essential to support accounting. While it is true that only few applications of videoconferencing would require individual authentication among participants, it is also true that most of them would require some mechanisms to support the service provider to charge users for the resources they use. Individual authentication is then required not only to prove the authenticity of each single party participating to the protocol, but also to support services such as access control and accounting, that otherwise would be more difficult to implement or to be accepted.

The interest towards applications using multicast protocols by the indistry is most of all motivated by the fact that these applications can have potentially a large market that is willing to pay for such applications (e.g. video/audio-on-demand, chat boards, bulletin boards, interactive on-line games). There is a lot of work about pricing and billing models [25], that can be adopted, and it is outside the scope of this paper to investigate advantages and drawbacks of each model. Rather, we say that these models may highly benefit from having a mechanism that is able to quantify how much of the system resources is used (e.g., bandwidth, memory, cpu time, etc), and most important by whom. Individual authentication allows then to provide an authentic and unforgeable account of the entity that has consumed a particular resource. Individual authentication is then necessary to support accounting for:

– *senders*, in case users are charged on the basis of the information they actively sent to the group.

– *receivers*, in case users are charged on the basis of the information they read from the group.

Accounting requires the user making the reservation to be authenticated, so that the correct user can be charged [3]. Accounting for reservations introduces a level of complexity to the Internet that has not typically been experienced with non-reserved traffic, and requires network providers to have reciprocal usage-based billing arrangements for traffic carried between them. It also requires mechanisms whereby some fraction of the bill for a link reservation can be charged to each of the downstream multicast receivers.

Billing mechanisms may rely on quite complex schemes for the reservation of resources. It is plausible a scenario where the user does not know exactly how many resources she needs to be able to use a particular service, the only thing she will specify is the type of service she wants and for which she is ready to pay. There could be then a reservation manager, that could serve one or more end-users, that will deal with the technical mapping between type of service and amount of resources required to provide it. Then the reservation manager, upon authorization by the end-user, may request to the service provider the necessary amount of resources on behalf of the end-user. In this kind of scenario, where all the parties involved, end-user, reservation manager and service provider may have conflicting interests, individual authentication is a necessary requirement in order to trace the behaviour of each participant (e.g., the end-user wants to be assured that she "really" uses all the resources the reservation manager required).

3.5 Sender Authorization and Denial of Service

Individual authentication and accounting are not only essential in order to support charging and billing, but also to support access control mechanisms. Effective authorization mechanisms have to rely on the fact that the hosts presenting the request of service/resource are really who they claim to be, often on an individual basis. Individual authentication makes it possible to write access control lists with a fine granularity thus allowing flexible and complex policies, not otherwise possible.

Because authorization is difficult, current multicast practice is also bound to a high risk of denial of service. Anyone can join a group and anyone can send to the group, at any time. Some broadcast-like sessions with quite large audiences have already been targeted by attacks, where unauthorized senders inserted their own transmission flow [19]. Typical configurations allow anybody who knows the multicast address, that is not secret, to inject information in the communication. To date, this is prevented by encrypting all the traffic with a single encryption/decryption key, distributed somehow to all the authorized members of the group. This however does not stop members to misuse the service without being detected. It also fails to trace which specific member had the key stolen.

4 Conclusion

As argued in each of the above subsections, the unicast security solutions that have become standard and well-accepted do not work well in the multicast network context and with respect to the multicast applications. The issues themselves are modified and normally more complex. This requires new research. But it also requires prototypes and working applications. In particular, the applications shown to work well with the MBONE should be extended to deal with some of the mentioned security issues. However, those solutions and implementations should be efficient and scalable, and should adapt to the evolution of protocols and the network and transport levels.

References

1. R. J. Anderson and C. Manifavas. Chameleon: a New Kind of Stream Cipher. In *Proc. Second Workshop on Information Hiding*, 1997. 126
2. G. Ateniese, M. Steiner, and G. Tsudik. Authenticated Group Key Agreement and Friends. In *ACM Conf. on Communications and Computer Security*, 1998. 126
3. F. Baker, B. Lindell, and M. Talwar. RSVP Cryptographic Authentication. *draft-ietf-rsvp-md5-08.txt*, February 1999. 120, 128, 129
4. A. Ballardie. Scalable Multicast Key Distribution. *RFC 1949*, May 1996. 126
5. I. Brown, C. Perkins, and J. Crowcroft. Watercasting: Distributed Watermarking of Multicast Data. *Manuscript*, February 1999. 127
6. B. Cain, S. Deering, and A. Thyagarajan. Internet Group Management Protocol, Version 3. draft-ietf-idmr-igmp-v3-01.txt, February 1999. 120, 121
7. R. Canetti and B. Pinkas. A Taxonomy of Multicast Security Issues. *draft-canetti-secure-multicast-taxonomy-00.txt*, May 1998. 125, 126, 127
8. S. Deering. Host Extensions for IP Multicasting. *RFC 1112*, August 1989. 120, 121
9. S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, 1990. 122
10. S. Deering, D. Estrin, D. Farinacci, M. Handley, A. Helmy, V. Jacobson, C. Liu, P. Sharma, D. Thaler, and L. Wei. Protocol Independent Multicast-Sparse Mode (PIM-SM): Motivation and Architecture. *draft-ietf-idmr-pim-arch-05.txt*, August 1998. 121
11. S. Deering, D. Estrin, D. Farinacci, V. Jacobson, A. Helmy, D. Meyer, and L. Wei. Protocol Independent Multicast Version 2 Dense Mode Specification. *draft-ietf-pim-v2-dm-01.txt*, November 1998. 121
12. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. *RFC 2362*, June 1998. 121
13. W. Fenner. Internet Group Management Protocol, Version 2. *RFC 2236*, November 1997. 120, 121
14. S. Floyd, V. Jacobson, S. McCanne, C.G. Liu, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. In *ACM SIGCOMM 1995*, 1995. 120
15. S. Floyd, V. Jacobson, S. McCanne, C.G. Liu, and L. Zhang. A Reliable Multicast Framework for Light-weight Session and Application Level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997. 124

16. R. Gennaro and P. Rohatgi. How to Sign Digital Streams. In *CRYPTO 97 - LNCS 1294*, 1997. 127
17. C. Griwodz, O. Merkel, J. Dittmann, and R. Steinmetz. Protecting VoD the Easier Way. In *ACM Multimedia 98*, 1998. 127
18. M. Handley. SAP: Session Announcement Protocol. *draft-ietf-mmusic-sap-00.txt*, November 1996. 120, 123
19. M. Handley, J. Crowcroft, C. Bormann, and J. Ott. Very large conferences on the Internet: the Internet multimedia conferencing architecture. *Computer Networks*, 31:191–204, 1999. 119, 129
20. M. Handley and V. Jacobson. SDP: Session Description Protocol. *RFC 2327*, April 1998. 120, 123
21. M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. *RFC 2543*, March 1999. 120
22. V. Hardman, M. A. Sasse, and I. Kouvelas. Successful Multiparty Audio Communication over the Internet. *Communications of the ACM*, 41(5):74–80, 1998. 119
23. P. Honeyman, A. Adamson, K. Coffman, J. Janakiraman, R. Jerdonek, , and J. Rees. Secure Videoconferencing. In *Proc. 7th USENIX Security Symposium*, 1998. 125
24. M. P. Maher and C. Perkins. Session Announcement Protocol: Version 2. *draft-ietf-mmusic-sap-v2-00.txt*, November 1998. 120, 123, 128
25. L.W. McKnight and J.P. Bailey (Ed.s). *Internet Economics*. MIT Press, 1997. 128
26. S. Mittra. Iolus: a Framework for Scalable Secure Multicast. In *ACM SIGCOMM*, 1997. 126
27. S. Paul, K. K. Sabnani, J. C. Lin, and S. Bhattacharyya. Reliable Multicast Transport Protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, 15(3), April 1997. 120, 124
28. T. Pusateri. Distance Vector Multicast Routing Protocol. *draft-ietf-idmr-dvmrp-v3-08.txt*, February 1999. 121
29. Ed. R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. *RFC 2205*, September 1997. 120, 128
30. L. Rizzo and L. Vicisano. A Reliable Multicast data Distribution Protocol based on software FEC techniques. In *Proc. of 4th IEEE Workshop on the Architecture and Implementation of High Performance Communication Systems (HPCS'97)*. Elsevier, 1997. 124
31. S. Micali S. Even, O. Goldreich. On-line/off-line digital signatures. In *Advances in Cryptology - Crypto 89*, 1989. 127
32. K. Savetz, N. Randall, and Y. Lepage. *MBONE: Multicasting Tomorrow's Internet*. IDG, 1996. 119, 122
33. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *draft-ietf-avt-rtp-new-03.txt*, February 1999. 120
34. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *RFC 1889*, January 1996. 120, 124
35. T. Speakman, D. Farinacci, S. Lin, and A. Tweedly. PGM Reliable Transport Protocol Specification. *draft-speakman-pgm-spec-02.txt*, August 1998. 120, 124
36. D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. *RFC 1075*, November 1988. 121

Issues in Multicast Security
(Transcript of Discussion)

Francesco Bergadano

Dipartmento di Informatica, Università di Torino, Italy

Actually I think the topic I'm going to talk about is well-suited for the first theme of the workshop which was security protocols for the entertainment industry. Then the workshop theme changed and I was told we were going to talk about auditable protocols. Suddenly I realised my last year's talk was just perfect for the theme of this year but since I couldn't repeat the same thing, I decided I would go with this, but there are some things that I think are also relevant for auditing.

This is going to be a general talk about issues in multicast security and multicast protocols, so I will mainly put forward problems not really specific solutions, although we're obviously waiting for specific issues.

The work is done in a project for two years in the University of Turin, we hired Cavagnino for two years and the paper is also co-authored with Bruno.

First I am going to try to convince you in one minute that multicast is important, because this is not obvious. Most, 99% perhaps more, of Internet communications today are unicast, so if multicast is important why are things as they are? But I think multicast is going to be more important in the near future. Then I'm going to talk briefly about the context we're talking about, and therefore I will have to say something about session management protocols and transport protocols in the internet multicast scenario because otherwise we're just going to talk about things that are too abstract.

And then of course we're going focus on security issues: the objectives of security work as relevant for an IP multicast, the obvious solutions that we may borrow from traditional communication security in the unicast case, then I will try to convince you that the most obvious techniques are not adequate for the multicast case, and then I will give a very brief survey of what's been done beyond the obvious techniques that do not work.

So this is the one minute to convince you that multicast is important. Well we all know the increasing multi-media content, we get more and more images, we wait longer and longer on the browser.

Bob Morris: Multicast?

Reply: OK, the question is, what is multicast. Multicast may transmit from one source to many sources at the same time. For example, I'm doing a form of multicast now because I'm talking to you all at the same time. You see, if I wait to do my talk for David Wheeler first and then to you – that's unicast, one at a time – it would take me much longer, so multicast saves time. OK so there's more multi-media content so saving time is more important because I'm sending more information, saving repetitions is more important.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 132–139, 2000.

Then, I think this is particularly interesting, there's an obvious increase in one-to-many traffic on the Internet. We used to use the Internet just for FTP and for sending mail to friends, now most use is of the broadcast type, people connect to CNN and to live websites and get lots of information so the information gets sent again and again. And many websites are proposing streaming all the end readers. So this is quite in place and growing.

This is more debatable: the Internet conferencing application, where it doesn't work very well, but could work better in the future. So there might be wide-spread use of these tools in the future. And these are all applications where multicast is obviously superior to unicast. If I had a sender here in green, and if I were to send information to all of these four recipients in unicast I would send the same information four times along this communication line, with multicast I send it only once. So that's the whole purpose, saving resources, but resources are saved both for the network but also for the servers because all the communication set-up has to be done only once.

How is a multicast session handled? After talking about sessions and transport then we'll come to security, but a little more patience. A multicast session is a complex thing, first of all it has to be created. You obviously do not have a standard TCP connection, because a TCP connection is between two parties, here you have many parties. So you have a session, the session has to be initiated and somehow you have to obtain an IP multicast address for that session, and that's already a problem because it has to be a new one. Then you have to have advertise the session, you have to tell people: look there is a new session, you can join in, and so on and so forth. There have been protocols proposed for session advertisement and session description. Once the session is set up, people can send, senders can come out and send to the group, so sending to a multicast group means just sending information to an IP class D address. So the interesting thing is, you don't know who you send to. That has security implications. Receivers join the group by sending messages to the routers, but the senders do not know that receivers have joined the group.

After it is set up the session is almost ready, the only thing that is missing is effective routing, because the routers must manage to deliver data to the recipients and this is very different from unicast. Maybe the distance vector, this routing protocol has been used in the past and we're now moving to protocol-independent routing.

So, after here, it all works, it's working in the mBone. Well one thing that is still missing today (except experimentally) is the possibility to reserve quality of service which is of course necessary if you want decent video media. But this is going to change in the future, and the standardization of the ideas has already devised protocols such as RSVP to reserve quality of service.

So let's put up a session. After you have a session up, you have to find good ways to transmit information from senders to receivers, and good can mean two things. For real-time data, such as audio-visual, good means timely, and without too much loss of packets. So it doesn't mean reliable, for a good audio conversation I have to get 99% of the packets in a timely manner, if I lose 1%

I don't care, 1% is OK. And before I discuss its security implication, integrity changes the use of this. For some other applications, for example multicast of text or of software or data of some kind where I need reliability, for these other media we have protocols that enforce reliability and these are still in the standardization phase.

So security, in this context. Well first, security is needed. Why, both because of the network configuration and because of the applications. Its main application is privacy. For example, if I have a business meeting over the Internet by some kind of media audio and text conferencing tool I may want privacy first. So encryption is important. Then there is an issue of intellectual property and protection because the typical thing you do with multicast is send data that is interesting to the public, for example, music, stuff like that, so work on watermarking and stuff like that has to be considered but it is different in the multicast case. Then we want, really want, authentication. Again, for meetings, conferencing, we may want authentication for the same reasons we want authentication in many other contexts. But in the case of a group, authentication has a different meaning because you have to distinguish between individual and group, and I will say a few words about that. And finally, but importantly, the network will somehow move to a context where resources are in some way accounted for, so it will be necessary to know who is using resources, who's reserved the resources, that of course has authentication, authorisation, and so on. So we need security, but security is difficult, more difficult than the traditional case.

Well, first of all, we don't have a connection. It's easy to put SSL on top of TCP because we have a connection, so you exchange keys and all, but here you don't have a connection, you send to many people, how do you establish a connection. Worse that that, with the current protocols you don't even know who the receivers are, and even worse, receivers can suddenly wake up and become senders. If there is a new receiver which comes up and decides to send stuff all it needs to do is join the multicast group with a multicast router and then start sending information to the IP address. And there is a scalability problem, when you have to distribute keys to one person it may be a problem, when you have many persons in the session it's a nightmare.

Bob Morris: What is reliability? Received, sooner or later, correctly?

Reply: Correctly not in the cryptographic sense. We had a communication, we received every single bit correctly in the correct order.

Bob Morris: In the correct order?

Michael Roe: You can put it back in the correct order.

Reply: So what is the obvious solution, we should go to somebody who knows something about cryptography and ask how we solve these problems. And the obvious solution is, use some kind of PKI, distribute session keys, and then all you send will be encrypted with that session key. To do authentication, you also need a MAC, or you can do the two together as we have learnt to do yesterday. So authenticate and encrypt your session key.

Why doesn't this work? Well, two problems, key management and the problem of individual authentication. Key management: you have a session key, you

have to send it to a thousand persons, that's not easy. Then some of these persons go away from the session, you have to revoke the session key, but you cannot, they already have the key, so you have to distribute the key again. This is a big problem, many people are working on it.

Second problem: individual authentication. By using one session key for the whole group you may produce MACs that authenticate you as a member of the group, but the MAC doesn't tell who is sending information within the group, so we have to distinguish the two cases. Many applications need individual authentication, and you need that individual authentication for all the network requirements, including accounting and billing of the network resources.

So now I say a few words about key management. You have three problems, key distribution, key refreshment when you want to change the key, and you need to find a way to exclude members from the group when that is needed. Key distribution: there has been a lot of work on this, the easiest way to do it is to have a key distribution centre somewhere in the network, and every time you set up a session the key distribution centre sends the session keys. This is not scaleable. A better solution has been proposed in two RFCs – it is known as the group key management protocol – and in that case it is very similar to the obvious solution we just described, except that when you set up a session, at that time you identify a group key distribution centre, a group controller. Then the group controller will distribute session keys to each individual member, either by using PKI or by using long term shared keys. That's also not very scaleable.

A better, more scaleable, solution has been proposed in another RFC where, and that's a typical situation in multicast security, you use the network for some of the security things. We normally are not happy with using the network for security and even IPSec that we have heard about does encryption at the network level, so everything goes encrypted between the routers and therefore you have some disadvantages such as the fact that traffic analysis is impossible, but it is an advantage that you don't have to trust me. But in multicast, in many cases you have to somehow trust the method because otherwise you wouldn't do it, so for example in this proposal, the scaleable multicast redistribution, routers are delegated to distribute keys.

A nice proposal that has been done by Mittra (SIGCOMM 1997) is to have basically the same principle except you don't have the routers distribute the keys, you have some group intermediaries at a higher level in the protocol stack. And the interesting thing is, that group intermediaries use a different key for their subgroup, so they have to decrypt and re-encrypt which is of course a waste of time, but the nice thing is then member revocation is much easier, because member revocation can be dealt with at a local level just by the one group intermediary that is responsible for the member to be revoked. And then there is work that you can also use the concept of broadcast encryption that was proposed years ago by Fiat-Nauer where you somehow distribute session keys in such a way that member revocation is easier. And there's also been much recent work on extensions of Diffie-Hellman to third parties. In ACM SIGCCS 98 there's three papers on group key management, for example. These are all

things that are useful for key management, but there is no one good standard solution.

Mark Lomas: You said that you need to re-encrypt with the previous scheme, but if you change the session key regularly then you don't need to. The originator changes session keys regularly, and distributes them to the group controller, but the thing that would otherwise have to have done the redirection then says, I've changed this, change and tell your recipients that this is the new session key. And if you wanted to do revocation, don't tell them the new session key.

Reply: There is a solution in one of the protocols as I described, but then you have to redistribute the key, and if you have many users it can be awkward. For example with broadcast encryption you won't need to do that, you change the keys but you don't need to redistribute.

Other problem: individual authentication. How can you do an individual authentication in a multicast group? You can do it in two ways. Either you share keys with every member and you have as many MACs as you have members, which is not very good, both because key management is a problem and because their authentication data becomes very long. Or you do digital signatures, of course that gives individual authentication, but digital signatures are less efficient, especially you have to consider that multicast applications already load your PC a lot. I can easily get my CPU to 100% use, because of bigger compression and all the rest you have to do. If on top of that you have to do digital signatures, it can be awkward. Also if it has to be done on the fly by routers because of billing it's also a problem. Lots of this research may be applied not to computers but to set-top terminals, which are used especially in Europe. So efficiency is a problem, digital signatures are not a good solution.

Michael Roe: You don't have to check the signature on every packet though. You can imagine protocols whereby the unit that got authenticated was several multicast-IP packets, you only check the signature when you reach the end of the unit.

Reply: That's not good for billing, after the packet has gone.

So what kind of solutions can you use? For individual authentication there's not much research in this area though it's important. You can use a special kind of signature that is more efficient, such as Gennaro and Rohatgi. Or there has been a kind of protocol designed explicitly for this problem, where you use one-time signatures with chaining. It's a possible solution but signatures become very long. There's also been proposals for using symmetric cryptography, so that you may do authentication with many MACs but not as many as there are participants, a finite number. For example, I may have a thousand participants but I only use ten or a hundred MACs and that's fixed. This case has to do with coalition of corrupt receivers.

This is an area that I am working on and I think there is not much work done but it may become very important if we are going in the direction of billing some of the Internet resources, because in the end somebody will have to pay

for the resources. There's different ways of using the resources and you have to pay in a different way.

There's lots of work on ideas concerning reservation. You may reserve the resources with the reservation protocol known as RSVP, and people know that security is a problem here. For example – this is an Internet Draft dealing with RSVP authentication – when you reserve a resource you must say who you are. And also for the session announcement protocols, there is standardisation for authentication. When you create a session you must tell who you are, when you delete the session you must be the same person who created the session, denial of service is a problem if you don't.

So there is work on security for reservation-creating sessions, but surprisingly then you have no means for forward authentication. You must authenticate yourself when you start up things but then when you use things you don't do it because it's difficult. But of course it's necessary, otherwise you have to look at a concept of theft of service. This is a new-key word maybe, somebody else reserves traffic, somebody else creates the session, and then I jump in, the session has been paid for by somebody else and I use it, and you need individual authentication to solve this problem.

Protection of content is an intellectual property problem. Of course it's better to encrypt data if you want to protect it, but if you want to avoid having authorised receivers duplicate the data then you have somehow to do something else. But traditional watermarking is not good, because when you watermark a CD, for example, you watermark it differently for every recipient. Here you are broadcasting basically the same content to everyone, so the naive understanding of watermark is not good. So there's a number of people working on this, I think the Chameleon is relevant for that because as far as I understand, you send the same content but you change keys so that the decryptor can't understand it otherwise.

There is this new concept of *watercasting* introduced by John Crowcroft recently, where you use the network again to change the content while it's being distributed, so that you can watermark content again. There is also work where you multicast data that is not useable and then you unicast forward error correction so that it can be seen, but then the watermark is inserted. That's an interesting area.

Conclusion. Security is needed for multicast, and security's difficult for multicast, these are my main two points. And that's a very good thing for us researchers. But research is not enough. All this key management work is frustrating if you don't get a good solution. You get a lot of papers but you don't get a good solution.

So we also need prototype implementations. I think the research we are doing has to consider the precise context we're working on and the precise protocols that are being used, because it is very likely that you have to exploit the characteristics of multicast protocols to find your good solution. This is my last slide describing the project we're working on. We took the mBone application setting, with a session distribution interface, video, audio, a whiteboard, we shared

a whiteboard and a network text editor as distributed by UCL and others so we had the source code for all that, we kept it working with cameras and mainly on Sun workstations. We are currently working on certain means for efficient individual authentication with our kind of software. We did the project for two years and we expect the protocol soon.

Matt Blaze: It seems to me there are two things. There's multicast the networking primitive, which is how you efficiently distribute packets around the network to a subset of the network. And then there is multicast the security primitive, which is a question of how do you send the data to some out of all the people who might be able to receive the cypher text. Now it seems to me these are two separate problems, or it might be useful to think of them as two separate problems, since multicast the networking primitive seems to be of somewhat dubious utility. It's very hard, which has attracted a lot of researchers, people have been working on it for a while, but so far as we can tell there's really no commercial demand for it, it may or may not go anywhere, and many of the applications you describe, except for the real time ones where there are actual broadcasts going out at a particular time, can be addressed by more conventional web-caching style interfaces. A very useful principle then in designing protocols using multicast as a security primitive would be, to find a way to decouple them from multicast the networking primitive. Would some of the things you're thinking about work, for example, in a hierarchy of web caches?

Reply: I must say that I disagree with you, although what you say is perfectly reasonable. What I was trying to say is precisely that we should not think abstractly about security and then years later realise that there is a protocol in a network context that we have to deal with. But let me answer your question. Streaming video and audio, yes it's real time, so they're multicast infrastructure. For web pages you are suggesting you could use proxies instead of a multicast, but from a network perspective that's not a smart thing to do, it's a very ad hoc solution. First, you have to configure manually, it may work, it's basically having stated routing at the application level you have a stated logic at the application level.

Matt Blaze: Well there's no fundamental reason you have to do web caching manually and statically. The advantage of the caching approach is that it doesn't actually involve tinkering the network at all.

Reply: Yes, but in order to use your networking it doesn't take that much to change, CISCO routers have multicast protocols in them now, you only have to open the parameters. They do it, but most people don't. So most of the Internet doesn't do multicast, but the hardware supports it.

Stewart Lee: But not the security protocol.

Reply: Not security, no, and not the quality of service, that's a big problem.

Matt Blaze: I don't want to turn this into a networking workshop (laughter).

Bruce Christianson: The other linkage that Francesco is making is to tie the end security issues in to the question of resource usage in the network. That's another link that we would lose if we regard the two issues separately. Security isn't just not being able to read the text that you're not entitled to read, it's

also not being able to reserve buffer slots and bandwidth unless you're a person who's entitled to receive the content, and this view makes a firm link that we usually miss between those two things.

Reply: Yes, I think there is a number of applications in security that are often not studied, maybe reliability, we were talking about a thermometer, maybe the reliability of the thermometers is more important than the privacy.

Bob Morris: Are you sending messages and you don't know who you're sending them to?

Reply: Unless you put something secure on the top of it like we were talking about then, at the network level, no, you don't.

Bob Morris: You say that CISCO will do what you need right now.

Reply: CISCO has routing protocols in the hardware. Usually system managers don't turn them on. So yes, the building of strategies is a big thing.

Michael Roe: But, for example, the CISCO we use does have that turned on, and we do use this. One of the reasons why the commercial Internet tends not to turn this facility on is because of security worries, and sometimes they don't dare do it because of the possibility of very large amounts of traffic being generated. Interestingly, when you talk about Internet traffic, you currently are billed for stuff you receive whether you wanted it or not.

Stewart Lee: Speaking as a user, I do not wish to be billed for what is sent to me, I wish to be billed for what I choose to receive.

Larry Paulson: I am trying to come to grips with this notion of multicast and I guess people like the real world metaphor. If they subscribe to the Times or something, then they just get a subscription somehow and then a copy of the Times is sent to them and to how many million other people, and it's really so many individual copies being given out and not multicast in your sense. And to have your subscription you don't go to the manager or some hierarchical thing, but it's just between you and the people who publish it. Now when one looks at this networking concept, either you have a million unicasts which isn't really multicast, or else you are somehow forcing people into a hierarchy which they are not going to be used to. So if I was to subscribe to some Internet service and say, well I want to buy this thing but I don't want to buy that thing, part of your problem is packaging it in such a way that the end user isn't less powerful.

But once you've got this hierarchy of things, it seems the problems involved in multicast involve one of the higher level nodes, and that there would be exponentially fewer of them than of the end users, so it would probably be a much more manageable experiment. One could imagine a combination of keys to each individual for example.

Reply: Yes, some of the solutions I cited are based upon a combination of keys.

Performance of Protocols
Extended Abstract

Michael Roe

Centre for Communications Systems Research

1 Introduction

NIST is currently running a contest to select a block cipher which will be the new "Advanced Encryption Standard". Many people have contributed to the evaluation of the candidate algorithms; however, much of the evaluation has concentrated on how fast the algorithms are, rather than other factors (such as their security) which might be more important.

This paper was presented at a security protocols conference, rather than a cryptography conference. Here the tradition is that we take a step back from discussing the internals of cryptographic algorithms and consider instead the context in which the algorithm is used. I think we should do this for discussions of performance as well as discussions of security. To the end user, what matters is the performance of the whole application. Is it worthwhile for cryptographers to spend a great deal of effort on micro-optimizing block ciphers? Will this have a significant effect on overall system performance?

2 Changes in Technology

When I started doing research in cryptography, in the mid 1980's, performance of cryptographic algorithms was certainly an issue. The typical workstation was a Micro Vax connected to a dumb terminal: the CPU wasn't fast enough to support a windowing system. The cryptographic algorithm in use was DES, which was designed for hardware, not software implementation. Although we now know many tricks for doing DES fast in software, these weren't widespread in the mid-80's. A consequence of all this was in was worth spending some effort optimizing the cryptography.

CPUs are now much faster. Memory and network speeds have also increased, but not nearly as much as CPU speeds. Pure computation, such as is used in a block cipher, is cheaper in both absolute terms and relative to other tasks, such as writing the data to disc. Unlike DES, nearly all of the AES candidates are designed for high performance in software.

It could be argued that for most applications, nearly all the AES algorithms are fast enough. We have reached the point where cryptography is not a significant portion of the total CPU burden, and the relative speed of the algorithms no longer matters very much. The purpose of this paper is to collect some of the evidence for and against this argument.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 140–146, 2000.

3 The Effect of APIs

When comparing the performance of two different cryptographic algorithms, we have to be very careful that we are comparing like with like. It is often the case that different implementations have different interfaces. (The AES reference implementations are an exception to this, as a common interface was defined). Surprisingly, the choice of interface matters. The overhead of the procedure call is quite large relative to the total amount of work the block cipher does, so the speed of the call matters.

At this point, protocol considerations intrude. An API that gives very fast performance in a benchmark may turn out to be useless in a real application, because it doesn't provide features which are needed:

- Often, we would like to have several keys in use at the same time. For example, different threads of control within a server might be talking to different clients, using different keys. This means that the key schedule (or a reference to it) should be a parameter to the encryption routine, not a global variable. This can effect both the time taken for the call itself, and the time taken for each access to the key schedule which the block cipher: Passing parameters can slow down the call, and indirect addressing can be slower than direct addressing.
- Often, we need to support dynamic negotiation of which algorithm to use. The block cipher of choice changes over time, and it is useful to be able to incrementally upgrade distributed systems one machine at a time. During transition you have to negotiate whether to use the old or the new algorithm. Export control and other political reasons mean that different clients of the same server may support different algorithms: again, dynamic negotiation is needed. In practice, this means that the call to the encrypt routine itself is indirectly addressed: it is entered by a jump to a subroutine whose address is held in a register, rather than jump to a subroutine at a fixed address. Such indirect calls are slower. More to the point, on many processors they disrupt the pipelining to a much greater extent that normal subroutine calls.

Block size also matters: a 128 bit block cipher will typically look faster than a 64 bit block cipher simply because half the number of subroutine calls are needed to encrypt the same amount of data – the subroutine call is a significant portion of the cost of the cryptography. An API that encrypts many blocks in one call will achieve greater performance than one that encrypts a single block per call.

Some machines are "big endian" and others are "little endian". Achieving interoperability between the two often involves exchanging the order of bytes within a word. Although some processors have a special instruction for doing this, it is typically not accessible from a high-level language. Like the subroutine call, the byte swap is a significant portion of the total CPU time. Some APIs attempt to hide the byte swap by pushing it outside the block cipher; they define the cipher as an operation on sequences of 32-bit words. Of course,

the protocol implementation as a whole still ends up doing the byte swap, but benchmarks that only measure the cryptographic algorithm are unfairly inflated by excluding it.

Memory management strategies also make a huge difference:

- Service provider allocates, nobody deallocates.
  The memory leak is a common implementation strategy – it does not give good performance, at least not if you need to make a large number of calls. Memory accesses are much more expensive than register accesses; paging is even worse.
- Service consumer allocates and deallocates.
  The typical problem with this is that the caller doesn't know how much to allocate – a protocol machine that doesn't know which algorithm is in use doesn't know how much space to allocate for a key schedule.
- Service provider allocates and deallocates.
  From a system point of view this is good, because the provider at least knows how big the buffers need to be. The downside is that it can result in far more calls to the memory allocator: one per key change, or even one per cipher block. Memory allocation is much more expensive than doing the actual encryption.

4 Chaining Modes

The choice of chaining mode can have surprising performance implications. In CBC mode, buffers need to be extended to a whole number of blocks on encryption and reduced to the real length on decryption. This may result in heap management calls which cost far more than the actual cryptography. Increasing the length of a buffer may require calls to heap management routines for the simple reason that the extended data won't fit, and a bigger buffer must be allocated. Reducing the length of a buffer may also cause heap management calls. This may come as a surprise to C programmers, as the usual C memory allocator remembers the original length of each object on the heap, even its size has subsequently changed. Memory allocators used by other languages can be less forgiving, and may need to be notified when a buffer's size changes.

As a result, length preserving modes such as Output Feedback (OFB) can give better performance than CBC mode. Although the number of encryption operations is roughly the same, fewer calls to heap management routines are needed.

5 Some Performance Measurements

This section presents two sets of performance measurements: firstly, the traditional cryptographic benchmark of the block ciphers in isolation, and secondly measurements of the block ciphers in a particular application.

The block ciphers were all implemented in the Ada programming language, rather than C. Many of the reasons for this choice of programming language are outside the scope of this paper – they were implemented for use by a project which I won't describe here. However, for the purposes of interpreting the benchmarks, it is important to realise that care needs to be taken when comparing them with other measurements of algorithms implemented in C. Such comparisons run the risk of confusing differences between the cryptographic algorithms with the differences between the compilers that were used.

However, relative comparison of these benchmarks is more meaningful: each implementation is in the same programming language, with the same API, the same coding style, and the same compiler used.

Figure 1 shows the speed in Mb/s of several applications, as measured by a program which encrypts a million 128-bit blocks under the same key. The machine used for measurement was a 266Mhz Pentium II running Linux. The compiler used was GNAT 3.11p at optimization level 3 with array bounds checking disabled. The test program does very little else apart from encrypt data; for example, it doesn't write the ciphertext to disk or an IO device. As it doesn't do anything between each encryption, most of the encryption routine will still be in the primary cache. This is a common set-up for measuring algorithm performance; however, it is not necessarily typical of what happens in real applications.

Algorithm	Encrypt (Mb/s)	Decrypt (Mb/s)
RC6	40.6	36.0
Rijndael	27.5	26.1
CAST 256	22.8	22.6
Serpent	11.9	10.6
Safer+	3.56	3.52

Fig. 1. Bulk Encryption Speed

The above table shows that there is nearly an order of magnitude difference in performance between the slowest and fastest algorithms, when they are measured in isolation.

Figure 2 shows the time taken to perform a remote procedure call with three different levels of cryptographic protection: no cryptography; data integrity provided by a MAC based on SHA-1; integrity and confidentiality provided by a MAC based on SHA-1 followed by encryption with RC6 in cipher block chaining mode. The RPC implementation used was GLADE. (Ada includes remote procedure calls as part of the specification of the programming language. GLADE is an implementation of these Ada RPCs). Two different types of remote procedure call were measured: one which takes an integer parameter and returns an integer result, and one which takes a string parameter and returns a string result. For timing the string RPC, both the parameter and the returned value

were 45 bytes long. We expect the string RPC to take longer than the integer RPC because more data must be transmitted. This measurement was also taken with block ciphers other than RC6, but these are not shown in figure 2 because the differences between algorithms is less than the accuracy of the measurement. RC6, Rijndael and CAST 256 all result in an integer RPC taking 2.7 mS to two significant figures. This in itself is enough to provide evidence for the contention that was made at the beginning of this paper: the differences between cryptographic algorithms are small compared to the total amount of work, at least for all applications which use this particular RPC library.

Cryptography	Integer RPC (ms)	String RPC (ms)
None	1.2	1.5
SHA-1	2.5	2.8
SHA-1 + RC6	2.7	3.1

Fig. 2. Time taken for a remote procedure call

In order to measure the differences between algorithms, a more sensitive experiment was devised. The time taken for 10,000 remote procedure calls was measured 8 times. Figure 3 presents the mean, upper quartile and lower quartile of the distribution of the results. The purpose of presenting the upper and lower quartile is to give an indication of the accuracy of the measurement; other events taking placing within the same computer system can effect the timing, and it is important to separate this background noise from differences due to the choice of algorithm. In figure 3, the actual timings have been divided by 10,000 to give a figure in milliseconds per RPC.

A comparison of figures 1 and 3 shows that the performance of block ciphers measured in isolation gave a reasonably good prediction of their performance in the RPC application. Ciphers which were fast in isolation were also fast in the RPC application.

With the integer RPC, the call fits into 5 128-bit cipher blocks and the return value fits into 4 cipher blocks, so each RPC requires 9 encryptions and 9 decryptions. With the string RPC, the call fits into 8 blocks and the result into 7, resulting in 15 encryptions and decryptions. The actual timings in figure 3 are very close to those we would predict by converting the speed results of figure 1 into microseconds per block and multiplying by the number of blocks which are encrypted. One exception to this is the Serpent algorithm, which did rather worse in the RPC application than would have been predicted from its performance in isolation.

A remaining question is how much of computational cost of a protected RPC is due to computing a SHA-1 hash. Figure 4 shows the time in milliseconds needed to compute the SHA-1 hash of messages of various lengths. Very short (1 byte) messages take longer than 16 byte messages, because internally SHA-1

Integer RPC			
Cryptography	Mean	Lower quartile	Upper quartile
SHA-1 + RC6	2.6660	2.6658	2.6661
SHA-1 + Rijndael	2.6997	2.6985	2.7013
SHA-1 + CAST 256	2.7172	2.7168	2.7176
SHA-1 + Serpent	3.0563	3.0557	3.0569
SHA-1 + Safer+	3.1466	3.1460	3.1470
String RPC			
Cryptography	Mean	Lower quartile	Upper quartile
SHA-1 + RC6	3.0534	3.0534	3.0536
SHA-1 + Rijndael	3.1214	3.1210	3.1217
SHA-1 + CAST 256	3.1403	3.1398	3.1403
SHA-1 + Serpent	3.5594	3.5590	3.5598
SHA-1 + Safer+	3.9573	3.9410	3.9414

Fig. 3. Time taken for integer and string RPC

pads messages to a whole number of blocks, and this padding process takes a measurable amount of time. Apart from this effect, the time taken to compute a hash is approximately a constant amount plus an amount proportional to the size of the message.

Size (bytes)	Time (mS)	Speed (Mb/s)
1	0.0153	0.523
16	0.0156	8.21
32	0.0158	16.2
48	0.0160	24.0
64	0.0253	20.2
80	0.0263	24.3
128	0.0360	28.4
256	0.0573	35.7
512	0.100	41.0
1024	0.186	44.0
2048	0.357	45.9
16384	2.76	47.5
65536	11.04	47.5

Fig. 4. Time taken to compute SHA-1 hash

Comparison of figures 2 and 4 shows that for small messages the cost of computing the hash function is on the order of 1% of the cost of performing an RPC. This implies that replacing SHA-1 with a faster hash function will have little effect on the total time needed to perform a cryptographically protected RPC.

Comparison of figures 3 and 4 shows that although SHA-1 is faster than any of the AES block ciphers for large messages, some block ciphers are faster for short messages. In the RPC application, messages are short. When one of the faster block ciphers is in use, a CBC MAC computed using a block cipher would be a faster means of providing integrity than the MAC based on hash functions that was actually used. A third way of providing integrity is to compute a hash of the message, and then protect this hash using a CBC MAC. These results show that for short messages it is quicker to compute the MAC directly on the data.

6 Conclusions

These results show that for applications using a particular RPC library, the computational cost of encryption is small compared to the total computational cost of performing an RPC. We conjecture that this is also true for many other applications. A consequence of this is that a cipher designer should not further sacrifice speed for security in order to make their design look faster than the competition. Further improvements in block cipher performance will have negligible effect on applications as a whole.

Unfortunately, it is not the case that adding cryptographic protection has little impact on application performance. Adding cryptography nearly doubled the time taken to perform an RPC. However, the additional computational cost is not primarily due to the cryptographic primitives, the hash function and the block cipher. Rather, it is is due to the cost of adding another layer to the protocol stack, with the associated additional multiplexing, buffer management and header parsing. This effect is well known to opponents of the OSI seven layer model: additional layers often mean additional overhead.

Much work has been done on fast software implementation of block ciphers. These results show that this effort is misplaced. A greater effect on application performance could be achieved by developing fast implementations of the security protocol itself, not the block cipher on which it relies.

Performance of Protocols
(Transcript of Discussion)

Michael Roe

Centre for Communications Systems Research

These are some thoughts I had based on the Advanced Encryption Standard contest that NIST has been running recently. NIST has been asking people to submit their ideas for crypto algorithms in an open evaluation process where everybody has a look at the algorithms and submits their comments. If this was a crypto conference I would be going into great detail on the internals of block ciphers. But this is not, this is a protocols conference and what we generally do is we try to take one step back. Rather than looking at the internals of what's going on inside round functions, we start thinking about the application and what actually the block cipher is for. In the context of this AES contest, I think all the cryptographers have been looking at the wrong thing. Everybody's been optimising block ciphers to shave small numbers of clock cycles off the time it takes to encrypt a block. OK, we are familiar with this. When I started doing work in this field in the mid 80s, the machine on my desk was a MicroVAX. It was connected to a dumb terminal, and it didn't run the X windows system, firstly because Xwindows didn't exist at that point, and even if it did the machine was too slow to run it. On this machine, doing encryption was a substantial proportion of the total amount of CPU the application used. And so consequently it was actually worth going to the trouble of hand assembly coding the block cipher to make it some tiny bit faster.

In the years since then things have changed. CPUs have got much, much faster. Memories have also got faster but not to the same extent, and applications have become excessively bloated. As a consequence, the time a block cipher is taking is no longer a large proportion of the total time. It's really insignificant. So all this assembly code optimising and choosing block cipher X over block cipher Y because it's three milliseconds faster is really neither here nor there. Looking at this I'm going to make some remarks on where the CPU time really is going.

Most of the places where the CPU is really going are in the application. I was recently building something that encrypted voice. So you take voice, digitise it, GSM compress it, encrypt it, and send it down the phone line. Then you look at where the time is going. The GSM audio compression, to get 64 kilobits per second of digitised voice down to 13 kilobits per second, is far more CPU intensive than the small amount of block encryption. Similarly, a lot of my work is dealing with streams of compressed video, and doing MPEG-2 compression on a video stream is again far more computationally intensive than encrypting it. Even if you don't look at application functions like compression, you just look at the crypto protocol, you discover that even that takes more time than the block cipher.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 147–152, 2000.

One surprising thing is that the API's actually matter. The cost of getting from your protocol code into the cipher and out again is quite comparable with the CPU time it spends in there just doing the round function a couple of times. So that, for example, the 128 bit block ciphers that we're now seeing in the AES always look faster than 64 bit block ciphers because you do 128 bits at a time. You're doing half the number of subroutine calls into the subroutine that does the business. It looks twice as fast because the subroutine calls dominate and you're doing half as many of them.

Another point that's worth thinking about is buffer management. If you've ever implemented communications protocols, you'll know a lot of stuff you're doing is allocating buffers, copying data from network interfaces into them, and freeing the buffers again. Copying and freeing buffers is actually a computationally expensive business, particularly because you're touching memory which is not in the primary cache. It turns out that your cipher modes matter. Things like cipher block chaining increase the size of the message slightly. You have to pad out the message to a full number of blocks, and then on decryption you suddenly discover there's padding in the block so you have to shrink it. Does it matter? If you're just writing down protocols you might think this doesn't cost anything. When you're actually implementing this, you realise that all this business has cost you more than doing the crypto. So a thing as simple as whether you use output feedback mode or cipher block chaining mode can have a huge performance impact for reasons that are nothing to do with the choice of the block cipher.

Stewart Lee: I think that what you say is absolutely correct, but I think that it's a story we've seen before many times. I can remember spending hours optimising the speed with which the square root routine runs, or the sine routine runs or whatever, and it took much longer to get into the routine than it did to actually perform the routine. This has been going on for a long time in other circumstances and I'm sure what you're saying is absolutely accurate, that we spend a lot less time encrypting than we do getting there, as it were, but it's nothing that surprises me.

Bruce Christianson: There's another point of view, which is the extent to which you can actually share things like buffer resources. There are some things that you might be doing in buffers with cryptography that mean you really don't want to release those buffers in the state that you leave them in. So sometimes you've got to put things back into a clean state after you've done all that you have to do, which costs you a lot of effort; or you've got to maintain separation between two buffer pools, which means there's lots of copying; or you just don't think about it and have a nice insecure application.

Ross Anderson: There are always going to be applications where people design right up to the wire. The ones that have been talked about mostly at the AES conference are smartcards and other peanut processors. For example, if you were doing a standard VISA-style electronic purse protocol, the customer's card and the merchant's card between them do 14 encryptions and decryptions and because none of them has any memory to speak of they have to do 14 key setups

as well. With some algorithms such as E2, key setup takes three times as long as encryption and certainly it falls below the line of acceptable performance, even though there are better algorithms in the AES process. You are pushing it some to get that transaction done in under a quarter of a second. So there's that kind of application. Another kind of application was the one that Craig ... of DigiTel talks about, where basically he can persuade his bosses to put in crypto provided that he uses less than 1% of the CPU, because they are trying to do so many other things as well. There, given the data rates involved, even though they're using beefy processors it still does make a difference at the margin. Brian Snow made an interesting comment at the workshop. People do design right up to the wire, and if Moore's Law continues for another thirty years and people feel the need to upgrade from 128 bit keys to 192, then many people may find that they can't because they don't have the performance, they don't have space in the protocols or the packets or whatever. So there's many performance related issues that come along, some of them being quite unobvious at first sight.

David Wheeler: I think there's a case for dealing with larger segments at a time. I think a variable block length code avoids many problems of choice. If the protocol uses, say, DES, you have to use a chaining mode or something. If the application said code 20 words and send it, that protocol choice would be removed, resulting in a simpler system. According to you it might also be faster.

Reply: I agree there are applications where performance is critical. One is where you're doing IP-level encryption in encrypting routers. There are two things that are particularly important about this: you're an intermediate system rather than an end system, and you're not doing the work of the whole application. The end system may well be doing video compression, but as a router in the middle of the network, you don't see that, you just see uninterpreted data and your job is to shovel it in one end and out the other as fast as possible. For a firewall encrypting router, that additional cost of doing encryption is a substantial added burden. I saw some recent measurements of IPSEC where turning on the cryptography reduced performance by 13%. Those applications where you're doing almost no processing of data other than cryptographic processing still exist, but I think they're in the minority now.

Matt Blaze: I think in the case of the encrypting routers, that's partially an artefact of the way that virtually all the router vendors have, under enormous pressure, put in IPSEC during the last part of the design process. Whereas the code for routing and processing packet headers and so on is very carefully optimised as essential to the core part of their business. So this might just be an artefact of the way IPSEC has been been engineered into specific router implementations rather than something fundamental.

Reply: That's probably true. Let's say that routers are one of the few pieces of software that are fully optimised. A lot of effort has been put in to making them fast, so then adding crypto shows. Whereas if you put it into something like your typical web browser then the slowness of the rest of it hides the fact that the crypto slows things down a bit. But yes, I agree.

Dieter Gollman: There's a PhD. student at Royal Holloway who is implementing protocols that use crypto. He has done it on the current Gem Plus card, and he's done it on the Schlumberger card. Communications with the card take ten times as long as the cryptography, and he's already cut down his data used to something like 400 bytes, which is the basic communication unit of his protocols. His observation is that at that stage API's matter. The way you communicate with the card matters much more than the cryptography on the card.

Virgil Gligor: One comment about the progress that we made in performance. Information compression has reached some limits which information theory tells us about, and people have looked at compression for a long, long time, perhaps much more intensively than cryptography. I still feel that it's worth paying attention to the performance of the cryptographic algorithms and modes and so on, because it turns out that there is a lot of fat to cut. It may not be the best place to put your efforts, maybe it's the second best place, but there is some progress that can be made there. Whereas I think that compression, unless technology changes a lot, and information theory changes a lot, is not as profitable to work in because the breakthroughs are going to be very, very, sparse.

Wenbo Mao: [Some] authentication protocols using cryptographic algorithms include a loop to slow it down, to prevent it being used for bulk encryption — because authentication uses strong encryption.

Bruce Christianson: Very often you're not in the position of somebody saying "Here's my algorithm, now can you make it run faster?", but rather somebody saying "I want some security, how much can I have? I've got a particular part of my bandwidth or part of my CPU that I'm willing to devote to all this stuff, how much security can I have?"

Reply: I think one of the lessons is that having more CPU to spend on your crypto doesn't actually buy you anything at all. Your're probably going to need an extra buffer copy or a subroutine call to do any crypto at all, and once you've paid that you can have as strong crypto as you like and it costs you next to nothing extra.

Larry Paulson: A curious thing with some backup software I just bought. It has the option of compressing your backup and also encrypting it. The encryption slows things down tremendously.

Markus Kuhn: That's because drives in PCs tend to be unbuffered, so the entire I/O has to go through the CPU. There's no asynchronous DMA involved.

Ross Anderson: Stewart said that somebody made a survey of systems over a period of decades and came to the conclusion there is about one bit of I/O per CPU cycle.

Stewart Lee: This was a statistic that was gathered in the days of the early 360s, that one instruction was executed for one bit of I/O that was done. It was common across a wide range of machines. The only times where machines got computationally intensive is in situations where the Jet Propulsion Lab was rendering some pictures from outer space to make whatever information it con-

tained visible and then it got up to about three quarters of a bit of I/O for each instruction executed. Or it got down to that.

Ross Anderson: Well in that case you're in trouble because block ciphers like the typical AES candidates are taking about two cycles to encrypt one bit.

Stewart Lee: I'm just reporting on what happened. The thing came up because IBM was trying to sell us some faster machine with the same I/O gear and channels and so on and I'd say they just wanted to charge us for waiting faster.

David Wheeler: But actually the cost in MIPS of installing a security system is a very small proportion of the total cost. There's cost in human effort, administration and all these other things which we haven't added in the equation.

Reply: The cost of actually getting the key there securely in the first place is the real stumbling block of deployment in most of these things. People can tolerate a 5% reduction in speed, but having to type in some obscure key into a pop up window whose function they don't understand can cause the whole thing to completely fall at the first hurdle.

Virgil Gligor: I think that the point you are making is that we pay too much attention to the performance of the crypto box itself and not enough to the system level issues of using crypto, and I think that's a very valid point. RC6 inside SSLeay is not the same RC6 you are measuring. It's a different RC6 in terms of performance.

Reply: Yes, there's the algorithm negotiation effect. Everybody's favourite algorithm changes over time, and there's export control. This means that a single server may be talking to clients that use different algorithms, either because they're a mixture of old and new clients that have a different choice of algorithm, or domestic and foreign clients that use different crypto. Either way, you need to do a dynamic switch on the algorithm identifier in the protocol, which means that in really low-level machine terms there is probably going to be a jump to the contents of a register rather than a jump to a fixed address. This can have a different effect on the pipelining on some processors. But we've got to do this kind of dynamic algorithm negotiation. Merely having the choice that you could have picked something else other than RC6 makes your RC6 go slower, because you're having to go through the dynamic stuff. My measurements show that it adds about 1 microsecond per block.

Virgil Gligor: Simply using one of these cryptographic service providers inside a framework such as Microsoft's CAPI is going to make it a lot more expensive. The routing of the calls would be through the framework itself, what you call the APIs, is going to cost a lot.

Reply: Yes, sure, what I was trying to say was even if you've optimised it to make it as efficient as possible there are some things they fundamentally have to do. They're going to slow you down quite a bit. But yes, using some of these frameworks means additional unnecessary inefficiencies on top of the necessary inefficiencies.

Ross Anderson: I'd say it was worse than that. The biggest screwup I think that NIST made when writing the AES call was insisting on a 128 bit block,

because 64 bit blocks are so entrenched, in all the protocols used in banking and applications like that, that it just can't be replaced. It's not economically feasible to do so. So you're not only going to have to support different algorithms, you're going to have to support different block lengths.

Reply: Implementing the different key sizes was also most entertaining. I did some experiments, putting the AES candidates in to some software applications. Having a variety of choices of key size causes problems all over the system. You've got to push it all the way up to some administrator level where somebody gets to choose which one they're having today, and having interfaces all the way through the system to get that information there really does cost in terms of changing the system that surrounds the cipher.

Ross Anderson: So there's going to be a requirement in US government systems that everything have mechanisms built in so that somebody can push a button to cause all 128 bit keys to be replaced by 192 bits. The engineering costs of that are going to be awesome.

Reply: Yes. There's another reason I didn't like the 128 bit blocksize. Imagine you have an application which wants both integrity and confidentiality and has short messages, such as yes or no, whether a particular event has or has not happened in the last minute, and so on. You compute a MAC on the message and then CBC encrypt it, so it becomes three blocks in length, probably to protect one bit of application data. If the block size goes from 64 bits to 128 bits, you've doubled the size of every message, as it's still three blocks. So your communications bandwidth has just doubled, and in some applications you can't tolerate that.

Ross Anderson: It's going to be much worse than that because the typical crypto APIs make separate calls to do the MAC and to do the encryption, and this is one of the reasons that the banking network is essentially unprotected.

Integrity-Aware PCBC Encryption Schemes

Virgil D. Gligor* and Pompiliu Donescu

VDG Inc., 6009 Brookside Drive, Chevy Chase, MD 20815
{gligor,pompiliu}@eng.umd.edu

Abstract. Traditional encryption schemes, such as Cipher Block Chaining (CBC), are unable to detect integrity violations caused by adaptive chosen-message (i.e., chosen-plaintext *and* ciphertext) attacks when used with typical non-cryptographic Manipulation Detection Code (MDC) functions, such as bitwise exclusive-or, modular addition, CRC-32, and quadratic checksums. In this paper, we define secure Plaintext-Ciphertext Block Chaining (PCBC) schemes that detect such violations at a low performance cost, thereby preserving both message secrecy and integrity against chosen-message attacks. We present the salient properties of these schemes, their security, and preliminary performance measurements.

1 Introduction

It is well known that secure, block-based encryption schemes, or modes, do not necessarily preserve message integrity. Equally well-known is the need to protect message integrity at a low performance cost [7,26]. Attempts to achieve this with block-based encryption schemes have typically relied on non-cryptographic Manipulation Detection Codes (MDCs), particularly on checksums such as CRC-32, bitwise exclusive-or, quadratic functions, and modular addition [11,15,22,8,19]. Further, non-synchronizing encryption schemes, which provide error propagation to the end of a message, have been proposed for achieving both message secrecy and integrity in a single processing pass by appending some simple redundancy, such a constant-filled block or just the first plaintext block, to *the end* of the plaintext before encryption [17,19]. The checksums and redundancy functions that can be computed in a parallel or pipelined manner are of particular interest. We henceforth refer to functions producing them as the high-performance Manipulation Detection Code (hpMDC) functions. (A more detailed, though informal, definition of hpMDCs is provided by Gligor [12].)

Encryption schemes that require two, possibly concurrent, passes over the data to provide message integrity include those that compute a keyed Message Authentication Code (MAC) of the input using a different key from that used for encryption, and send the MAC along with the encrypted message [19]. They

* This work was performed while this author was on sabbatical leave from the University of Maryland, Department of Electrical and Computer Engineering, College Park, Maryland 20742.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 153–168, 2000.

also include those that apply a keyed hash function to the input to be encrypted and then encrypt the result of this function along with the input [8,22]. Unless the computation of the keyed MAC or hash function is performed concurrently with message encryption, most of these schemes are unlikely to (1) approach the performance of schemes that protect message integrity with hpMDCs for most message sizes, and (2) be applicable to real-time applications where commencing verification of message integrity cannot be deferred till the end of message decryption [20]. Also, two-pass schemes are less suitable for implementation in low-power, hardware devices than schemes that use hpMDCs.

Typical hpMDC functions, such as those for computing bitwise exclusive-or, modular addition, CRC-32, and quadratic checksums, or for providing a constant-filled block to be appended to the end of the message plaintext before encryption, cannot be used with traditional encryption modes, or schemes, to detect integrity violations caused by adaptive chosen-message (i.e., chosen-plaintext *and* chosen-ciphertext) attacks. The most general of these attacks enable an adversary to forge ciphertext messages that would be decrypted correctly with non-negligible probability by an unsuspecting party. (We use the notion of negligible probability in the same sense as that of Naor and Reingold [18].) The adversary need not know, nor be able to predict, the plaintext produced by correct decryption of the forged ciphertext. Less general, but nevertheless potent, attacks are considered successful only if correct decryption of forged ciphertext with non-negligible probability would also cause the resulting plaintext to contain predictable, known, and even chosen, values. These values would allow an attacker to verify *a priori* whether the integrity checks would pass with non-negligible probability. An example of such an attack against the Cipher Block Chaining (CBC) mode of encryption when used in conjunction with the CRC-32 – one of the strongest checksums in wide use [22,8] – in which the adversary can predict the plaintext of a forgery is provided by Stubblebine and Gligor [24]. Other block encryption schemes that are susceptible to such attacks when using the typical non-cryptographic MDCs include the original CBC [19] and Plaintext Cipher Block Chaining (PCBC) modes [17,16,12], and the XOR schemes [2][1].

In this paper, we define new single-pass PCBC encryption schemes that maintain message secrecy and integrity against the most general adaptive chosen-message attacks at a low performance and implementation cost. We present the salient properties of these schemes, their security, and preliminary performance measurements.

2 Integrity-Aware PCBC Schemes

In this section we define new PCBC encryption schemes and their integrity properties. In defining an encryption scheme, we adopt the approach of Bellare *et al.* (viz., [4,2]), who show that an encryption scheme can be viewed as the triple (E, D, KG), where E is the encryption function, D is the decryption function,

[1] Unlike the original CBC and PCBC schemes, the design of the XOR schemes was not intended to allow integrity checking with typical MDCs.

and KG is the probabilistic key-generation algorithm. The encryption scheme is implemented with a block cipher, which can be modeled with secure finite families of pseudorandom functions (PRFs) or pseudorandom permutations (PRPs). In this context, a finite family of functions, F, consists of a set of functions and a set of strings (i.e., the set of keys), each string identifying a member function, f. Each function f maps $\{0,1\}^l$ to $\{0,1\}^L$, where l/L denotes the input/output length, and hence we say that finite family F has input/output length l/L. Let R be the set of all functions that map $\{0,1\}^l$ to $\{0,1\}^L$. The finite family F is *pseudorandom* if the input-output behavior of a function $f = F_K$, which is identified by key K drawn uniformly at random from the set of keys, "looks random" to someone who does not know K [4].

2.1 Definition of the iaPCBC Schemes

In the encryption schemes presented below, the key generation algorithm KG outputs a random, uniformly distributed, k-bit key K for the underlying PRP family F, thereby specifying functions $f = F_K$ and $f^{-1} = F_K^{-1}$ of l-bits to l-bits; similarly, when two distinct keys K, K' are used in a scheme, $f' = F_{K'}$ and $f'^{-1} = F_{K'}^{-1}$. The plaintext message to be encrypted is partitioned into a sequence of l-bit blocks (padding is done first, if necessary), $x = x_1 \cdots x_n$. Throughout this paper, $\oplus$ is the *exclusive-or* operator and $+$ represents *modulo* 2^l *addition.*

The encryption schemes defined below use a separate *block chaining sequence* for the encryption of each message, much in the same way as the ciphertext blocks are used in the traditional CBC schemes. The block chaining sequence, which is denoted by r_i, $i = 1, \cdots, n$, is defined as follows.

Sequence Value. The value of each element of the block chaining sequence, r_i, is computed as $r_i \leftarrow r_{i-1} + x_{i-1} + y_{i-1}$, where x_{i-1} is the plaintext and y_{i-1} is the ciphertext of the $i-1$ block, $i = 1, \cdots, n$, and is not revealed outside the encryption schemes. Other functions, or combinations of functions, not just the addition modulo 2^l of r_{i-1}, x_{i-1} and y_{i-1}, could be used to define the block chaining sequence r_i (e.g., subtraction modulo 2^l). The choice of such functions must produce a block chaining sequence r_i that (1) is secret (unlike the block chaining sequence of the traditional CBC mode which consists of ciphertext blocks), (2) allows proofs of the resulting scheme's security against chosen-message attacks, and (3) can yield fast implementations in software or hardware.

Sequence Initialization. In *stateless* implementation of the iaPCBC schemes, $r_0 \leftarrow \{0,1\}^l$; i.e., r_0 is initialized to a new random, uniformly distributed, l-bit value for every message. In contrast, in *stateful* implementations, r_0 is initialized to a new random, uniformly distributed, of l-bit value for every key, K, and a per-message counter is used to generate a new value of r_0 for every message. In both types of implementation, y_0 is initialized to $f(r_0)$, and x_0 is initialized to a new pseudorandom (and thus, uniformly distributed) l-bit value for every message that is independent of r_0 and, just as r_0, remains secret. In practice, the initialization of x_0 may be performed by encrypting a function of r_0 with f; e.g., $f(r_0 + 1)$. Other functions, not just addition modulo 2^l of one to r_0, can be used for generating x_0. It is important, however, that the encryption of these

functions of r_0 produce a pseudorandom value for x_0 that is independent of r_0, and remains secret.

Sequence Initialization with a Separate Key. In both the stateless and stateful schemes defined below, the values of y_0 and x_0 may be generated by encryption with a separate key, K', that differs from that used for encryption of the rest of the message, namely K. Hence, the function f' selected by the new key K' would be used only for generating $y_0 = f'(r_0)$ and $x_0 = f'(r_0 + 1)$, and f would be used for the encryption of the rest of the message blocks. This two-key variant of the encryption schemes defined below improves security against exhaustive key-search attacks (discussed in Section 3 below).

Stateless iaPCBC Scheme (iaPCBC$)
The encryption and decryption functions of the stateless scheme, $\mathcal{E}-iaPCBC\$^{F_K}(x)$ and $\mathcal{D}-iaPCBC\$^{F_K}(z)$, are defined as follows.

function $\mathcal{E}-$iaPCBC$\$^f(x)$
$r_0 \leftarrow \{0,1\}^l$
$y_0 \leftarrow f(r_0);\ x_0 = f(r_0 + 1)$
for $i = 1, \cdots, n$ **do** $\{$
$r_i \leftarrow r_{i-1} + x_{i-1} + y_{i-1}$
$y_i = f(r_i \oplus x_i)\ \}$
return $y = y_0 || y_1 y_2 \cdots y_n$

function $\mathcal{D}-$iaPCBC$\$^f(z)$
Parse z as $y_0 || y_1 \cdots y_n$
$r_0 \leftarrow f^{-1}(y_0);\ x_0 = f(r_0 + 1)$
for $i = 1, \cdots, n$ **do** $\{$
$r_i \leftarrow r_{i-1} + x_{i-1} + y_{i-1}$
$x_i = f^{-1}(y_i) \oplus r_i\ \}$
return $x = x_1 x_2 \cdots x_n$

Stateful iaPCBC Scheme (iaPCBC)
The encryption and decryption functions of the stateful scheme, $\mathcal{E}-iaPCBC^{F_K}(x, r_0)$ and $\mathcal{D}-iaPCBC^{F_K}(z)$, are defined as follows.

function $\mathcal{E}-$iaPCBC$^f(x, r_0)$
$y_0 \leftarrow f(r_0);\ x_0 = f(r_0 + 1)$
for $i = 1, \cdots, n$ **do** $\{$
$r_i \leftarrow r_{i-1} + x_{i-1} + y_{i-1}$
$y_i = f(r_i \oplus x_i)\ \}$
$r_0 \leftarrow r_0 + 2$
$y = y_0 || y_1 y_2 \cdots y_n$
return (y, r_0)

function $\mathcal{D}-$iaPCBC$^f(z)$
Parse z as $y_0 || y_1 \cdots y_n$
$r_0 \leftarrow f^{-1}(y_0);\ x_0 = f(r_0 + 1)$
for $i = 1, \cdots, n$ **do** $\{$
$r_i \leftarrow r_{i-1} + x_{i-1} + y_{i-1}$
$x_i = f^{-1}(y_i) \oplus r_i\ \}$
return $x = x_1 x_2 \cdots x_n$

Note that in the iaPCBC scheme $r_0 \leftarrow \{0,1\}^l$ is a parameter of the sender's state and, unlike parameters x and y, is secret.

2.2 Integrity

The most common method used to detect modifications of encrypted messages applies a MDC function (e.g., a non-keyed hash function [19]) to a plaintext message and concatenates the result with the plaintext before encryption. Upon receipt of an encrypted message, the message is decrypted and accepted only after the integrity check is passed; i.e., after decryption, the concatenated value of the MDC function is removed from the plaintext, and the check passes only if

this value matches that obtained by applying the MDC function to the remaining plaintext [11,19,12]. This technique has been used in commercial systems such as Kerberos V5 [22,24] and DCE [8,24], among others.

The iaPCBC schemes are intended to use hpMDC functions for message-integrity protection in the conventional way described above by appending the output of such a function, $g(x)$, *to the end* of the message plaintext. More formally, the resulting encryption scheme ($m\mathcal{E}-iaPCBC\$^{F_K}(x)$, $m\mathcal{D}-iaPCBC\$^{F_K}(z)$, KG) is defined as follows:
$y = m\mathcal{E}(x) = \mathcal{E}-iaPCBC\$^{F_K}(x||g(x))$, where $||$ is the concatenation operator, and
$x = m\mathcal{D}(z) = \mathcal{D}-iaPCBC\$^{F_K}(z))|| - x^r(\mathcal{D}-iaPCBC\$^{F_K}(z))$,
if $g(\mathcal{D}-iaPCBC\$^{F_K}(z)||-x^r(\mathcal{D}-iaPCBC\$^{F_K}(z))) = x^r(\mathcal{D}-iaPCBC\$^{F_K}(z))$, where $x^r(\mathcal{D}-iaPCBC\$^{F_K}(z))$ represents the decryption of the function $g(x)$ output within the decrypted ciphertext $\mathcal{D}-iaPCBC\$^{F_K}(z)$, and is a m-bit long string that fits into $r = \lceil m/l \rceil$ blocks of the message x; $\lceil\ \rceil$ is the ceiling function, and $||-$ is the function that removes string $x^r(\mathcal{D}-iaPCBC\$^{F_K}(z))$, from the decrypted ciphertext $\mathcal{D}-iaPCBC\$^{F_K}(z)$; or
$x = Null$,
if $g(\mathcal{D}-iaPCBC\$^{F_K}(z)||-x^r(\mathcal{D}-iaPCBC\$^{F_K}(z))) \neq x^r(m\mathcal{D}-iaPCBC\$^{F_K}(z))$, where $Null$ is a failure indicator. Functions $m\mathcal{E}$ and $m\mathcal{D}$ use the standard padding of block encryption schemes. Function g may also use padding independent of $m\mathcal{E}$ and $m\mathcal{D}$.

The goal of the encryption scheme defined above is to protect message secrecy and integrity against adaptive, chosen-message attacks. These attacks are defined by a protocol between an adversary A and an oracle O as follows.

1. A and O select encryption scheme
   $(m\mathcal{E}-iaPCBC\$^{F_K}(x), m\mathcal{D}-iaPCBC\$^{F_K}(z), KG)$,
   and O selects, uniformly at random, a key K of KG. A is allowed $q'' - 1$ encryption-with-hpMDC-generation queries, and one decryption-with-hpMDC-verification query. A must be allowed verification queries, or else it cannot test whether its choice of ciphertext is correct (since A does not have key K).
2. A sends encryption queries x^i, $i = 1, \cdots, q'' - 1$, to O. O flips a coin and, based on the result of the coin flip, it responds to A by returning either $y^i = m\mathcal{E}(x^i)$, $i = 1, \cdots, q'' - 1$ where x^i are A's chosen plaintext messages, or $y^i = m\mathcal{E}(\$^{|x^i|})$, $i = 1, \cdots, q'' - 1$, where $\$^{|x^i|}$ denote random, uniformly distributed, strings of the same lengths as those of x^i. (A conducts a chosen-plaintext attack against the iaPCBC\$-x_0 scheme in the real-or-random sense as defined by Bellare *et al.* [2].)
3. After receiving O's responses, A guesses ciphertext y where $y \neq y^i, i = 1, \cdots, q'' - 1$, and sends the decryption query y to O. (y is said to be not-previously queried (unqueried), or forged.) O returns $x = m\mathcal{D}(y)$ to A. Adversary A is successful if $x \neq Null$.
4. A and O execute the attack protocol for time t'' using memory μ'' for A's $q'' - 1$ encryption queries and one decryption query.

We say that the iaPCBC schemes are "integrity aware" with respect to a hpMDC function $g(x)$ in a chosen-message attack if the adversary's probability of success is negligible. When implemented with the CBC mode and used to encrypt messages consisting of an integer number of l-bit blocks (possibly after padding), the Variable Input Length (VIL) cipher of Bellare and Rogaway [5,6] can be shown to be integrity-aware with respect to several hpMDC functions including the bitwise exclusive-or, CRC-32, addition *modulo* $2^l - 1$, the selection of a single constant-filled block or just block x_1 of every message, whose output is appended *to the end* of the message before encryption [12]. However, the VIL cipher uses two sequential passes over its input and, thus, its performance is lower than those of single-pass schemes for all but very short messages.

To illustrate the integrity awareness of the iaPCBC schemes with respect to a hpMDC function, we choose $g(x) = x_0$, where x_0 is internally defined by both the iaPCBC\$ and iaPCBC schemes. In this example, block $g(x) = x_0$ is appended *to the end* of a n-block message plaintext x, and hence block $x_{n+1} = x_0$. For this choice of hpMDC function, the integrity check performed at decryption becomes $x_0 = f^{-1}(y_{n+1}) \oplus r_{n+1}$. An adversary is successful if the forged ciphertext produced in the attack defined above passes this check with non-negligible probability. Hence, an upper bound for this probability represents a quantitative measure of the "integrity awareness" of the iaPCBC schemes with respect to the choice of hpMDC function $g(x) = x_0$. The schemes obtained by the use of hpMDC function $g(x) = x_0$ are denoted by iaPCBC\$-x_0 (and iaPCBC-x_0) throughout this paper.

3 Properties of the iaPCBC Schemes

The iaPCBC schemes have notable secrecy and integrity properties in several areas.

1. Support for Message Integrity. The iaPCBC schemes require only a single cryptographic primitive, namely the block cipher that is necessary for encryption, to maintain integrity. Further, externally generated hpMDCs (i.e., those not generated as part of iaPCBC\$ or iaPCBC encryption), such as the CRC-32 and a zero-filled block appended to the end of a plaintext message (after padding, as necessary), can also be used with the iaPCBC schemes for protection against adaptive chosen-message attacks (unlike the original PCBC schemes [12]). Such functions would not offer a substantial performance or security improvement over the internally generated hpMDC function $g(x) = x_0$.

2. Support for Real-Time Message Authentication. The iaPCBC schemes can produce good Message Authentication Codes (MACs). We suggest without proof that y_n produced by the encryption of a string x of n blocks with the iaPCBC schemes is a pseudorandom function. (The proof of this follows along the same line as that of Bellare, Killian and Rogaway for the CBC-MAC [4].) Unlike the original CBC-MAC, the iaPCBC\$-MAC $= (y_0, y_n)$, and its stateful counterpart, thereby obtained is not vulnerable to splicing attacks [24,4]. Like the Double

MAC [20], both the iaPCBC\$-x_0 (iaPCBC-x_0) encryption and the stateless and stateful iaPCBC MACs are particularly useful for real-time message sources in which (1) the message length remains unknown until the message ends, (2) the beginning of message authentication cannot be deferred until the end of message receipt, and (3) only small, fix-sized, buffers for authentication processing are available, as would be the case with most hardware implementations.

3. Support for Multiple Encryption Modes. The definition of block chaining sequence of the iaPCBC schemes can be particularized to obtain other modes of encryption. For example,

- if $r_i = y_{i-1}$ and r_1 is the initialization vector, one obtains the original CBC scheme;
- if $r_i = x_{i-1} + y_{i-1}$ and r_1 is the initialization vector, one obtains the new version of the PCBC scheme proposed by Meyer and Matyas [17,19];
- if $r_i = r_{i-1} + x_{i-1}$, one simply randomizes each input block of an ECB encrypted message with a per message random value, r_1;
- if $r_i = r_{i-1} + y_{i-1}$, and r_0 is the initialization vector and $y_0 = f(r_0)$, one obtains a new integrity-aware CBC mode (defined for a weaker attack [14]).

In contrast with the other definitions of block chaining, the iaPCBC schemes capture the complete "history" of message encryption; i.e., in the encryption of every block, the chaining variable $r_i = r_{i-1} + x_{i-1} + y_{i-1}$ captures the entire message plaintext and ciphertext "history" beginning with the per-message initialization values r_0, x_0 and y_0 and ending with the previous block. This ensures that every block encryption includes this complete message "history" up to the current block. Intuitively, this diminishes the chances that splicing or decomposition of an iaPCBC-x_0 message would allow an adversary to produce a forged ciphertext that passes the integrity check. Conventional initialization-vector attacks [27] are also countered.

4. Support for Parallel or Pipelined Encryption. The choice of hpMDC function illustrated in this paper, namely $g(x) = x_0$, allows the parallel or pipelined implementation of the iaPCBC schemes. Other hpMDC functions would also allow such implementation, since they be executed in a parallel or a pipelined manner (by definition). For example, for parallel execution using $g(x) = x_0$, each plaintext message x is partitioned into L segments, $x^{(1)}, \cdots, x^{(L)}$ after customary block-level padding (n.b., this L should not be confused with the output length of a PRF, which is typically denoted by L, also). Each segment, $x^{(s)}, s = 1, \cdots, L$, consists of one or more l-bit blocks, and if $g(x) = x_0$ is used, then an additional l-bit block is included in each segment. Each segment is encrypted/decrypted in parallel on a separate processor.

In parallel or pipelined implementations of the iaPCBC schemes, the initialization and computation of the block chaining sequence is performed on a per-segment basis starting with a common value of r_0, which is the new random, uniformly distributed, l-bit value for every message (in stateless implementations) or for every key (in stateful implementations). Also, the per-message value y_0 is

computed as $y_0 \leftarrow f(r_0)$. The initialization of the block chaining sequence for message segment s can be $r_0^{(s)} = r_0 + s, x_0^{(s)} = f(r_0^{(s)} + L)$, and the block chaining sequence can be $r_i^{(s)} \leftarrow r_{i-1}^{(s)} + x_{i-1}^{(s)} + y_{i-1}^{(s)}$. In stateful implementations r_0 is updated to $r_0 + 2L$ after the encryption of each message. (Other functions, not just addition modulo 2^l, can be used for the initialization of $r_0^{(s)}$ and $x_0^{(s)}$, and for the computation of the per-segment, block chaining sequence.)

The encrypted segments of a message are assembled to form the message ciphertext. Segment assembly encodes the number of segments L, the length of each segment n_s and, implicitly, the segment sequence in the message (e.g., all can be found in the ASN.1 encoding). If the segments of a message have different lengths, segment assembly is also synchronized with the end of each segment encryption or decryption within a message.

At decryption, the parsing of the message ciphertext yields the message length, L, segment sequence number, s, and the length of each segment, n_s. Message integrity is maintained both on a per segment and message basis by performing the per-segment integrity check; if $g(x) = x_0$, the per-segment check is $x_0^{(s)} = f^{-1}(y_{n_s+1}^{(s)}) \oplus r_{n_s+1}^{(s)}$. Both the message length and the segment sequence number within the message are encrypted in every single block of every segment of a message, since both are encrypted in the per-segment value of x_0^s, which initializes chaining sequence of segment s of a message. This helps avoid additional message integrity checks beyond the per-segment check. Failure of any per-segment integrity check, which also detects out-of-sequence segments and message-length modifications, signals a message integrity violation.

We illustrate a parallel implementation of the iaPCBC schemes below.

Stateless Parallel iaPCBC Scheme (piaPCBC$)
The encryption and decryption functions of the stateless scheme, $\mathcal{E}\text{–}piaPCBC\$^{F_K}(x)$ and $\mathcal{D}\text{–}piaPCBC\$^{F_K}(z)$, are defined as follows.

function $\mathcal{E}$–piaPCBC$\$^f(x)$
partition x into L segments $x^{(s)}$
each of length n_s;
$r_0 \leftarrow \{0,1\}^l$; $y_0 \leftarrow f(r_0)$;
for segment $s, s = 1, \cdots, L$, **do** {
$r_0^{(s)} = r_0 + s$
$x_0^{(s)} = f(r_0^{(s)} + L)$
for $i = 1, \cdots, n_s$ **do** {
$r_i^{(s)} \leftarrow r_{i-1}^{(s)} + x_{i-1}^{(s)} + y_{i-1}^{(s)}$
$y_i^{(s)} = f(r_i^{(s)} \oplus x_i)$ }
$y^{(s)} = y_1^{(s)} \cdots y_{n_s}^{(s)}$ }
assemble $y = y_0 || y^{(1)} \cdots y^{(L)}$;
return y.

function $\mathcal{D}$–piaPCBC$\$^f(z)$
parse z into y_0 and L segments $z^{(s)}$
each of length n_s;
$r_0 = f^{-1}(y_0)$
for segment $s, s = 1, \cdots, L$ **do** {
Parse $z^{(s)}$ as $y_1^{(s)} \cdots y_{n_s}^{(s)}$
$r_0^{(s)} = r_0 + s$; $x_0^{(s)} = f(r_0^{(s)} + L)$
for $i = 1, \cdots, n_s$ **do** {
$r_i^{(s)} \leftarrow r_{i-1}^{(s)} + x_{i-1}^{(s)} + y_{i-1}^{(s)}$
$x_i^{(s)} = f^{-1}(y_i^{(s)}) \oplus r_i^{(s)}$ }
$x^{(s)} = x_1^{(s)} \cdots x_{n_s}^{(s)}$ }
assemble $x = x^{(1)} \cdots x^{(L)}$;
return x.

Stateful Parallel iaPCBC Scheme (piaPCBC)
The encryption and decryption functions of the stateful scheme, $\mathcal{E}{-}piaPCBC^{F_K}(x, r_0)$ and $\mathcal{D}{-}piaPCBC^{F_K}(z)$, are defined as follows.

function $\mathcal{E}-$piaPCBC\$$^f(x, r_0)$
partition x into L segments $x^{(s)}$
each of length n_s;
$y_0 \leftarrow f(r_0)$;
for segment $s, s = 1, \cdots, L$, **do** {
$r_0^{(s)} = r_0 + s$
$x_0^{(s)} = f(r_0^{(s)} + L)$
for $i = 1, \cdots, n_s$ **do** {
$r_i^{(s)} \leftarrow r_{i-1}^{(s)} + x_{i-1}^{(s)} + y_{i-1}^{(s)}$
$y_i^{(s)} = f(r_i^{(s)} \oplus x_i)$ }
$y^{(s)} = y_1^{(s)} \cdots y_{n_s}^{(s)}$ }
assemble $y = y_0 || y^{(1)} \cdots y^{(L)}$;
$r_0 \leftarrow r_0 + 2L$;
return (y, r_0).

function $\mathcal{D}-$piaPCBC\$$^f(z)$
parse z into y_0 and L segments $z^{(s)}$
each of length n_s;
$r_0 = f^{-1}(y_0)$
for segment $s, s = 1, \cdots, L$ **do** {
Parse $z^{(s)}$ as $y_1^{(s)} \cdots y_{n_s}^{(s)}$
$r_0^{(s)} = r_0 + s$; $x_0^{(s)} = f(r_0^{(s)} + L)$
for $i = 1, \cdots, n_s$ **do** {
$r_i^{(s)} \leftarrow r_{i-1}^{(s)} + x_{i-1}^{(s)} + y_{i-1}^{(s)}$
$x_i^{(s)} = f^{-1}(y_i^{(s)}) \oplus r_i^{(s)}$ }
$x^{(s)} = x_1^{(s)} \cdots x_{n_s}^{(s)}$ }
assemble $x = x^{(1)} \cdots x^{(L)}$;
return x.

Note that $r_0 \leftarrow \{0,1\}^l$ is a parameter of the sender's state and, unlike the call parameters x and y, is secret.

5. Resistance to Key Attacks. The iaPCBC schemes improve resistance to exhaustive key-table attacks that exploit the availability of a known constant (or any other known text) encrypted under different secret keys [27], to which other encryption schemes are vulnerable (e.g., the stateful XOR and CBC schemes [2]). This is the case because the block chaining sequences are *secret* and their use in the iaPCBC and iaPCBC\$ schemes confounds each message block individually with a value that is unknown to the attacker, in the same way as that of random counters [13].

Further, the iaPCBC schemes improve resistance to exhaustive key-search attacks that rely on verification of key guesses whenever a *separate key* is used for the initialization of block chaining sequences. This is the case because guesses of the key used for the initialization of block chaining sequences (e.g., the key used for encrypting r_0, to obtain y_0, and $r_0 + 1$, to obtain x_0) cannot be verified independently of the encryption key for message data (i.e., the key used for encrypting blocks $x_i, i = 1, \cdots, n+1$). For example, the guess of the secret block-chaining value $r_1 = r_0 + x_0 + y_0$ based on a guess of the message data key, K, using known/chosen plaintext-ciphertext pair (x_1, y_1), cannot be validated without knowledge of the key K' used for the initialization of the block chaining sequence. Discovery of r_0 and x_0, which would enable such validation since y_0 is known, requires knowledge of key K'. Conversely, validation of a guess for key K', which is based on the decryption of y_0 to obtain r_0 and x_0, would require knowledge of the block chaining value r_1, and hence of the the key K, (presumably based on the known/chosen plaintext-ciphertext pair (x_1, y_1)).

4 Security Considerations

In this section, we provide evidence for the security of the iaPCBC schemes against both adaptive chosen-plaintext and chosen-ciphertext attacks. We first demonstrate the security of the iaPCBC$ scheme against adaptive chosen-plaintext attacks. The theorems and proofs that demonstrate that the stateful scheme (iaPCBC) and the two-key variations are secure are similar to that for the iaPCBC$ scheme and, therefore, will be omitted.

Lemma 1 and Theorem 1 below are restatements of Lemma 16 and Theorem 17 respectively, which are presented in the full version of the Bellare *et al.* paper ([2]). The proof differences and all other proofs of this paper are available at *http://www.ee.umd.edu/faculty/gligor.html*.

Lemma 1 (Upper bound on the security of the iaPCBC$ scheme in random function model). *Let A be any adversary attacking $iaPCBC\$(R_{l,l})$ in the left-or-right sense, making at most q' queries, totaling at most μ' bits. Then, the adversary's advantage is*

$$Adv_A^{lr} \leq \delta_{iaPCBC\$} \stackrel{def}{=} \left(\frac{\mu'^2}{l^2} - \frac{\mu'}{l}\right)\frac{1}{2^l}.$$

The following theorem defines the security of the iaPCBC$ scheme against an adaptive chosen-plaintext attack when the iaPCBC$ scheme is implemented with a (q, t, ϵ)-pseudorandom function F. F is (q, t, ϵ)-pseudorandom, or (q, t, ϵ)-secure, if an adversary (1) spends time t to evaluate $f = F_K$ at q input points via adaptively chosen queries, and (2) has a negligible advantage bounded by ϵ over simple guessing in distinguishing the output of f from that of a function chosen at random from R.

Theorem 1 (Security of iaPCBC$ against Adaptive Chosen-Plaintext Attacks). *Suppose F is a (t, q, ϵ)-secure PRF family with block length l. There is a constant $c > 0$ such that for any q', the $iaPCBC\$(F)$ is $(t', q', \mu', \epsilon')$-secure in the left-or-right sense, for $\mu' = ql - q'l$, $t' = t - c\mu'$, and $\epsilon' = 2\epsilon + \delta_{iaPCBC\$}$ where $\delta_{iaPCBC\$} \stackrel{def}{=} \left(\frac{\mu'^2}{l^2} - \frac{\mu'}{l}\right)\frac{1}{2^l}$.*

The iaPCBC$ and iaPCBC schemes should really be analyzed assuming F is a PRP family (not a PRF family), and hence one needs to apply the results of Proposition 8 of Bellare *et al.* [2] to the results of Lemma 1 and Theorem 1.

Let $x = x_1 x_2, \cdots, x_n$ be a plaintext message, and let $g^F(x) = x_0$ where $x_0 = f(r_0 + 1)$. (Note: The family of functions $g^F(x) = x_0$ is (q, t, ϵ)-pseudorandom.)

Theorem 2 (Security of the iaPCBC$-$x_0$ against a Chosen-Message Attack). *Let the scheme $(\mathcal{E} - iaPCBC\$^{F_K}, \mathcal{D} - iaPCBC\$^{F_K}, KG)$ be the stateless iaPCBC scheme, which is secure against adaptive chosen-plaintext attacks for parameters $(q', t', \mu', \epsilon')$, and let $g(x) = x_0$, $g \in g^F$. For any forged ciphertext $y = y_0 y_1 \cdots y_n y_{n+1}$, the probability that y is accepted by $\mathcal{D} - iaPCBC\$^{F_K}$ decryption after $q'' - 1$ encryption queries totaling μ'' bits and taking time t'', is*

$$Pr[x_0 = f^{-1}(y_{n+1}) \oplus r_{n+1}] \leq 3/2^l + \epsilon''$$

where $q'' = q' + 1$, $\mu'' = \mu' \leq (q - n - 3)l$, $t'' = t' - c'rq' \leq t - c(n + 3)$, $r = 1$, *and* $\epsilon'' = \epsilon + \delta_{iaPCBC\$}$, c' *is a constant depending on the speed of* g *and* c *is a constant depending upon the speed of* f *and* f^{-1}.

A similar theorem can be stated for the $iaPCBC - x_0$ scheme.

5 Performance Considerations

The performance of the iaPCBC schemes is (1) minimally degraded in comparison to that of the original CBC scheme [9,2], and (2) superior to that of the original CBC schemes, and most other similar schemes, when message integrity is desired.

The iaPCBC schemes add the overhead of a single modulo 2^l addition and block encryption per message (i.e., for generating x_0) and two *mod* 2^l additions per block to the traditional CBC scheme with random initialization vectors (or, equivalently, with initialization vectors set to zero and a random number in the first plaintext block [22])[2]. In principle, the execution of one of the two modulo 2^l additions per block can always be overlapped with the block encryption function and, hence, at peak speeds the only perceptible overhead is that of a single modulo 2^l addition per block. This overhead compares favorably with that added at peak speeds by an original CBC scheme that uses any of the hash functions known to date to provide message integrity. In any case, the dominant performance factor is the throughput achieved by block encryption. An important advantage of the new schemes is that their performance scales up nearly identically with that of block encryption; furthermore, hardware implementations of the new schemes can make their added overhead imperceptible.

To illustrate the performance characteristics of the iaPCBC$ and iaPCBC$-$x_0$ schemes, we used the *SSLeay* library [25], and conducted some preliminary measurements on a Sun SPARC Ultra 10 IIi processor running the SunOS 5.6 operating system. The processor has a 333 MHz clock, 2 MB of external (off-chip) cache, and 16/16 KB of internal (on-chip) instruction/data cache. We used the version 4.2 of the native C compiler with the -xO2 optimization option. We used a lightly loaded machine for our measurements, and the throughput for each of the eight message sizes was generated by averaging the results of fifty runs.

[2] Note that traditional CBC schemes require the use of secret initialization vectors that are protected from arbitrary modification, and the most common way of satisfying both requirements is to use (pseudo) random initialization vectors. For this reason, the generation of the initial per-message random number is considered to be a common overhead to both the new and the traditional CBC schemes. As shown above, the stateful iaPCBC scheme, however, requires only that a separate random number be generated per key, not per message, thereby eliminating much of this common overhead of stateless schemes. The alternative of generating and cacheing values of r_0 as the system runs and ahead of their actual use may also help decrease the overhead of the stateless iaPCBC scheme.

Our implementation was influenced by the implementation of the SSLeay library on 32-bit processors. However, we were able to use 64-bit additions for the iaPCBC$ scheme. The addition uses the *unsigned long long type* (64 bits) for the r and y operands of $r+ = y$, but the operand r must later be unpacked into unsigned longs (32 bits) for use in the encryption scheme (i.e., the DES encryption works on an array of two unsigned long elements). Also, the y values that result from the DES encryption are then re-packed into the unsigned long long y for the subsequent modular 64-bit additions. Each packing and unpacking operation, which would be avoided in a 64-bit implementation, requires a bitwise *or* and a shift.

The throughput of the CBC, iaPCBC$, CBC-MD5, iaPCBC$-x_0 schemes implemented with DES is shown in Figures 1 and 2 below for samples of both large and small messages. The percentage gain in the throughput performance of the iaPCBC$-$x_0$ scheme over that of the CBC-MD5 scheme for these message samples is shown in Figures 3 and 4.

The results shown in these figures indicate that, in unoptimized software implementations,

- a substantial overall throughput improvement can be expected; e.g., between 30% and 120% for small messages and about 20% for large ones. In general, we expect a much better performance improvement for small messages. For these messages, the performance of the MD5 hash function (and that of most hash functions) is closer to that of DES than for large messages, and hence our use of the function $g(x) = x_0$ improves a much larger fraction of the overall throughput of encrypted messages. (However, throughput measurements for small-size messages shown in Figure 2 are susceptible to a large margin of error caused by the inability to offset operating system effects over a fairly small number of test runs. For example, for 1 KB messages, the throughput of iaPCBC$ appears to be higher than that of CBC and for 100 byte messages the throughput of $iaPCBC\$ - x_0$ appears to be higher than that of iaPCBC$.)
- the clear performance bottleneck is that of the DES-CBC and underlying DES block encryption. Figure 1 shows that the performance differences between the DES-CBC, DES-iaPCBC$, and DES-iaPCBC$-x_0 are almost imperceptible for mid-size and large messages. Given the large advantage of using the $g(x) = x_0$ over that of using $g(x) =$ MD5 or any other hash function in iaPCBC encryption, we expect that the gain in the performance of the iaPCBC$-$x_0$ over that of CBC-MD5, or CBC-any-hash-function (e.g., SHA-1 [10]), to be even more pronounced whenever high-performance, block encryption functions, such as RC5 [1] or RC6 [23], are used.

We also expect that further improvements can be derived from an assembly language implementation where optimal register allocation can be performed for both the block encryption functions and the iaPCBC schemes.

It should be noted that the implementation of the iaPCBC schemes can be performed in software, hardware, or software with hardware support. Implementations can be in general-purpose computers or in dedicated hardware and

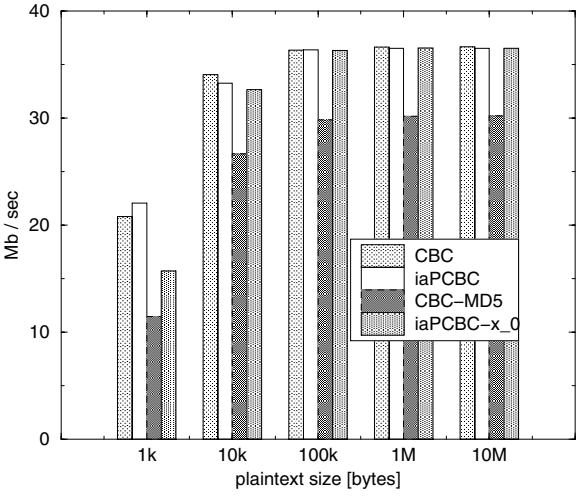

Fig. 1. Throughput of the CBC, $iaPCBC\$$, CBC-MD5, $iaPCBC\$ - x_0$ encryption schemes implemented with DES for message sizes of 1KB – 10 MB

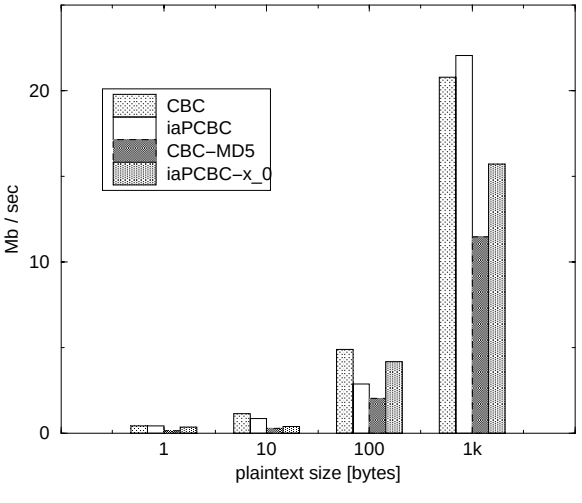

Fig. 2. Throughput of the CBC, $iaPCBC\$$, CBC-MD5, $iaPCBC\$ - x_0$ encryption schemes implemented with DES for message sizes of 1B – 1kB

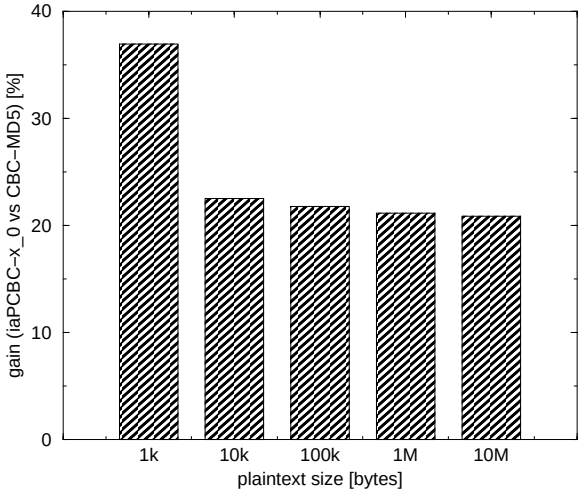

Fig. 3. Percentage gains of the DES-iaPCBC$-$x_0$ scheme over the CBC-MD5 scheme for message sizes of 1KB – 10 MB

software. The simplicity of the iaPCBC schemes suggests that substantial performance improvement can be expected when they are implemented in hardware. Of particular interest in this area are implementations of the iaPCBC schemes on low-power devices.

Acknowledgments

Radha Poovendran's suggestions helped improve the presentation of the proof of Theorem 2. Stuart Stubblebine co-authored several previous attempts to find integrity-aware CBC encryption schemes [14] with us. Mihaela Iorga and Serban Gavrila commented extensively on earlier notes on this topic. Doug Whiting provided an interesting example which shows that the two-key iaPCBC schemes do not counter all key-guessing attacks.

References

1. R.Baldwin and R.Rivest, "RFC 2040: The RC5, RC5-CBC, RC-5CBC-Pad, and RC5-CTS Algorithms," October 30, 1996. Available at *ftp://ds.internic.net/rfc/rfc2040.txt*. 164
2. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption," Proceedings of the 38th Symposium on Foundations of Computer Science, IEEE, 1997, (394-403). A full version of this paper is available at *http://www-cse.ucsd.edu/users/mihir*. 154, 157, 161, 162, 163
3. M. Bellare, R. Guerin, and P.Rogaway, "XOR MACs: New methods for message authentication using finite pseudo-random functions", Advances in Cryptology-

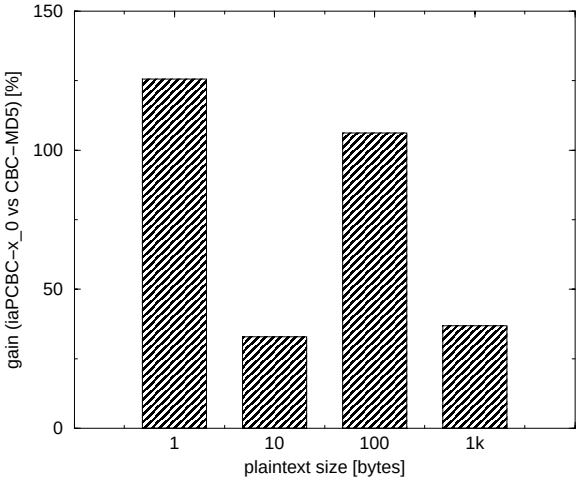

Fig. 4. Percentage gains of the DES-iaPCBC$-$x_0$ scheme over the CBC-MD5 scheme for message sizes of 1B – 1kB

CRYPTO '95 (LNCS 963), 15-28, 1995. (Also U.S. Patent No. 5,757,913, May 1998, and U.S. Patent No. 5,673,318, Sept. 1997.)

4. M. Bellare, J. Killian, and P. Rogaway, "The security of cipher block chaining", *Advances in Cryptology-CRYPTO '94* (LNCS 839), 341-358, 1994. 154, 155, 158
5. M. Bellare and P. Rogaway, "Block Cipher Mode of Operation for Secure, Length-Preserving Encryption," *U.S Patent No. 5,673,319,* September, 1997. 158
6. M. Bellare and P. Rogaway, "On the construction of variable-input-length ciphers," Proceedings of the 6th Workshop on Fast Software Encryption, L. Knudsen (Ed), Springer-Verlag, 1999. 158
7. S.M. Bellovin, "Cryptography and the Internet," *Advances in Cryptology - CRYPTO '98* (LNCS 1462), 46-55, 1998. 153
8. Open Software Foundation, "OSF - Distributed Computing Environment (DCE), Remote Procedure Call Mechanisms," Code Snapshot 3, Release, 1.0, March 17, 1991. 153, 154, 157
9. FIPS 81, "DES modes of operation", Federal Information Processing Standards Publication 81, U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1980. 163
10. FIPS 180-1, "Secure hash standard", Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, April 17 1995 (supersedes FIPS PUB 180). 164
11. V.D. Gligor and B. G. Lindsay,"Object Migration and Authentication," *IEEE-Transactions on Software Engineering*, SE-5 Vol. 6, November 1979. (Also IBM-Research Report RJ 2298 (3l04), August 1978.) 153, 157
12. V.D. Gligor, "Integrity Conditions for Symmetric Encryption," University of Maryland, Computer Science Technical Report, CS-TR-3958, December 1998 (revised April 1999). 153, 154, 157, 158

13. V.D. Gligor, "Symmetric Encryption with Random Counters," University of Maryland, Computer Science Technical Report, CS-TR-3968, December 1998. 161
14. V.D. Gligor, S.G. Stubblebine, and P. Donescu, "New Integrity-Aware CBC Encryption Schemes," University of Maryland, Computer Science Technical Report, CS-TR-3999, March 1999 (revised October 1999). 159, 166
15. R.R. Juneman, S.M. Matyas, and C.H. Meyer, "Message Authentication with Manipulation Detection Codes," Proc. of the IEEE Symp. on Security and Privacy, Oakland, CA., April 1983, pp. 33-54. 153
16. J. T. Kohl, "The use of encryption in Kerberos for network authentication", *Advances in Cryptology-CRYPTO '89* (LNCS 435), 35-43, 1990. 154
17. C. H. Meyer and S. M. Matyas, *Cryptography; A New Dimension in Computer Data Security*, John Wiley & Sons, New York, 1982 (second and third printings). 153, 154, 159
18. M. Naor and O. Reingold, "From Unpredictability to Indistinguishability: A Simple Construction of Pseudo-Random Functions from MACs," *Advances in Cryptology - CRYPTO '98* (LNCS 1462), 267-282, 1998. 154
19. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1997. 153, 154, 156, 157, 159
20. E. Petrank and C. Rackoff, "CBC MAC for Real-Time Data Sources," manuscript available at *http: //philby.ucsd.edu/cryptolib.html*, 1997. 154, 159
21. RFC 1321, "The MD5 message-digest algorithm", Internet Request for Comments 1321, R. L. Rivest, April 1992 (presented at Rump Session of Crypto '91).
22. RFC 1510, "The Kerberos network authentication service (V5)", Internet Request for Comments 1510, J. Kohl and B.C. Neuman, September 1993. 153, 154, 157, 163
23. R. L. Rivest, M.J.B. Robshaw, R. Sidney, and Y.L. Yin, "The RC6^{TM} Block Cipher," (submitted to the US Department of Commerce, NIST, for consideration as the new Advanced Encryption Standard (AES)), available at *http://theory.lcs.mit.edu/ rivest/publications.html* 164
24. S. G. Stubblebine and V. D. Gligor, "On message integrity in cryptographic protocols", Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy, 85-104, 1992. 154, 157, 158
25. SSLeay, available at *ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL* 163
26. J. D. Touch, "Performance Analysis of MD5," Proceedings of ACM, SIGCOMM '95, 77-86, 1996. 153
27. V.L. Voydock and S.T. Kent, "Security Mechanisms in high-level network protocols," *Computing Surveys*, 15(1983), 135-171. 159, 161

Integrity-Aware PCBC Encryption Schemes
(Transcript of Discussion)

Virgil Gligor

Department of Electrical and Computer Engineering,
University of Maryland

First let me say a few words about what I mean by integrity. Generally, we have a number of concerns in using crypto. One of them, is message authentication, in other words, knowledge as to whether the message comes from a particular source or from any source at all. We want to authenticate the content of messages as well.

Integrity is a different concern, which of course is required by authentication. Integrity says that what we would like to do is to detect all message modifications and possible forgeries with high probability. The minute I introduce the notion of high probability, and the notion of forgeries, I have an obligation to state what kind of attacks would lead to forgeries, and what kind of probabilities I'm talking about. So I shall do that a little later.

I'll cover just very briefly some of the traditional approaches to solving integrity problems. The goal should be clear throughout this presentation – what I'm interested is high performance and adequate security. I am not interested in absolute security with poor performance, although of course one can come up with encryption systems that have that property.

The problem of integrity has concerned me for a long time. My PhD dissertation identified a problem for further research. The problem was, how do we protect the integrity of abstract type objects once we move them outside the control of their type managers. Remember that at the time I was working on this problem, in the early and mid seventies, we all wanted to provide operating system support for abstract type objects. The problem that I noticed, and others noticed as well, was that if one backs up one of these abstract type objects on removable media, when objects came back to the system one has absolutely no guarantee of authenticity.

In late 1977 a friend of mine, Bruce Lindsay, who's been at IBM Research for a long time, and I, came up with up with what we thought at the time was great new idea. The idea was simply this: take an abstract object, add *some* redundancy to it and then encrypt it before exporting it. When the object comes back, decrypt the object and check the decrypted redundancy. We thought this would work with encryption in CBC mode, and we applied this technique, which we called *cryptographic checksums*, not only to abstract type objects but also to capabilities, which were also very difficult to move safely outside the realm of a particular system. By 1978 we wrote an IBM Research report and by 1979 we published a paper[1] on this subject. Notice what I just said, we add *some*

[1] IEEE Transactions on Software Engineering, vol. SE-5, No. 6, Nov. 1979, pp. 607-611.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 169–171, 2000.

redundancy to the object and we encrypt in CBC mode, because we naively believed that the CBC mode maintained the properties of the abstract type objects when *any* redundancy is present.

I completely forgot about the problem until 1991, when Stuart Stubblebine, who was a graduate student at the University of Maryland at the time, came up with a brilliant attack against CRC-32 used with DES-CBC encryption in Kerberos V5 proposal. The proposal was actually a confounded use of CRC-32. In other words, one adds a confounder, which is a random number prepended, to plain text. So there is no way one could discover the output of the CRC-32 because its input is first confounded and then encrypted in CBC mode. We presented Stuart's attack in a paper published at the 1992 IEEE Symposium on Security and Privacy, and we dropped the subject after a while.

Over the years, Stuart kept insisting that we could fix the CBC mode in some way and come up with a mode for which the CRC-32 would work. So we tried a few CBC changes and attacks against them last Summer, such as modifying in some way the cipher blocks before decrypting them. I tried to prove that those ideas work. Instead, I found out that none of our proposals worked. So last Fall I started working on a theorem about message integrity, and I'll try to show how one goes about proving the security *and* integrity of such encryption schemes.

What do people do generally when they talk about integrity of encrypted messages? Typically, they add some redundancy, they encrypt, then they decrypt the ciphertext that's provided to them, remove the redundancy and then check whether the redundancy function applied to the remaining plain text equals the decrypted redundancy value. The result of the decryption is rejected if this equation doesn't hold. This is a standard technique that's used in a lot of the systems that are in use nowadays, not only in Kerberos V5 and DCE.

Alternatively, what one can do is to encrypt a message with one key and then compute a MAC of the same message with another key, and present the two components as the output. This is, for example, what RSA does with some of their products. Now this is a secure scheme, but it's two-pass, so it doesn't satisfy my goal: namely, I want adequate security, which this provides, but I also want high performance, which this does not provide. No two-pass scheme will give you high performance, because you are going to have to go over the same string twice.

Michael Roe: That's not necessarily the case because you can compute both the CBC and the MAC simultaneously. The point we made is that they're not fundamentally serialised. You don't have to finish one of them before you start on the other of them. They are in principle parallelisable and good implementations will parallelise them.

An example of a protocol which I would say was genuinely in two parts is the one where you compute the MAC, put the MAC on the front of the message, and then CBC encrypt the whole thing, and that really is serialised because you can't start doing any encryption until you've done the whole MAC.

Reply: In my case, I wanted to choose a redundancy function g, which is extremely fast, such as just adding a block of zeros at the end of the message,

and then encrypt. One can't do much better than that. That's only a single pass. I'm suggesting a very high-performance manipulation detection code, which is the trivial one, just a block of zeros. In fact, there is at least one encryption scheme that already allows this.

In 1995 there was a series of measurements done which showed that MD5 had a throughput of about 50 megabits per second in a system where DES produced about 20 megabits per second. This showed that MD5 and SHA were faster, but not a lot faster (e.g., 10X) than block ciphers. This suggests that the hash functions that are used in place of redundancy function "g" could actually be expensive. Whereas if "g" is an XOR, one would use only very few operations per block. This difference doesn't materialise completely into throughput performance, but one can tell that it can make a big difference.

[*Editor's note:* At this point, the iaPCBC schemes are described. Two theorems are then presented regarding the chosen plain text security of the iaPCBC$ scheme and its integrity with a given redundancy function g(x). The second theorem contained a flaw, which was discovered later. For the details, see the the full version of the paper later in these proceedings.]

Several of the current encryption schemes have two *practical* problems. One: they do not preserve message integrity. Two: they produce a lot of known plain text. One can break their security wholesale under exhaustive key-table attacks. This degrades the value of the encryption scheme not just that of a key. In contrast, if someone is trying to exploit any known plain text in an exhaustive key-table attack against the iaPCBC schemes, they will fail regardless of how much known plain text they have. And this matters. If one goes to Bletchley Park, or reads Hinsley and Stripp's book[2], one notices that known plain text is deadly, not because of the exhaustive key-table attacks, but, as Mike [Roe] pointed out to me, because of other things that you might exploit.

Bob Morris: Known plain text is deadly.

Reply: Thank you, Bob. There is none here. None, not a single bit within the (pseudo)randomness of r_i.

Bob Morris: Not a single bit is just what you want.

[2] *The Codebreakers: the inside story of Bletchley Park*, Oxford University Press, 1993

The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks

Frank Stajano[1,2] and Ross Anderson[1]

[1] University of Cambridge Computer Laboratory,
New Museums Site, Pembroke Street, Cambridge CB2 3QG, UK
{name.surname}@cl.cam.ac.uk
[2] AT&T Laboratories Cambridge,
24a Trumpington Street, Cambridge CB2 1QA, UK
fstajano@uk.research.att.com

Abstract. In the near future, many personal electronic devices will be able to communicate with each other over a short range wireless channel. We investigate the principal security issues for such an environment. Our discussion is based on the concrete example of a thermometer that makes its readings available to other nodes over the air. Some lessons learned from this example appear to be quite general to ad-hoc networks, and rather different from what we have come to expect in more conventional systems: denial of service, the goals of authentication, and the problems of naming all need re-examination. We present the *resurrecting duckling* security policy model, which describes secure transient association of a device with multiple serialised owners.

1 Introduction

The established trend in consumer electronics is to embed a microprocessor in everything–cellphones, car stereos, televisions, VCRs, watches, GPS (Global Positioning System) receivers, digital cameras–to the point that most users have already lost track of the number of items they own that contain one. In some specific environments such as avionics, electronic devices are already becoming networked; in others, work is underway. Medical device manufacturers want instruments such as thermometers, heart monitors and blood oxygen meters to report to a nursing station; consumer electronics makers are promoting the Firewire standard [8] for PCs, stereos, TVs and DVD players to talk to each other; and kitchen appliance vendors envisage a future in which the oven will talk to the fridge, which will reorder food over the net.

We envisage that, in the near future, this networking will become much more general. The next step is to embed a short range wireless transceiver into everything; then many gadgets can become more useful and effective by communicating and cooperating with each other. A camera, for example, might obtain the geographical position and exact time from a nearby GPS unit every time a picture is taken, and record that information with the image. At present, if the

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 172–182, 2000.

photographer wants to record a voice note with the picture, the camera must incorporate digital audio hardware; in the future, the camera might let him speak into his digital audio recorder or cellphone. Each device, by becoming a network node, may take advantage of the services offered by other nearby devices instead of having to duplicate their functionality.

1.1 Ad-hoc Wireless Networks

This vision of embeddable wireless connectivity has been in development for several years at AT&T Laboratories Cambridge in the context of the Piconet [4] project and is also being pursued, although with emphasis on different aspects, by several other groups including HomeRF [6,12], IrDA [3] (which uses infrared instead of radio) and Bluetooth [13,7].

Everyone–including potential users–knows that wireless networking is more prone to passive eavesdropping attacks. But it would be highly misleading to take this as the only, or even the main, security concern.

In this paper we investigate the security issues of an environment characterised by the presence of many principals acting as network peers in intermittent contact with each other. To base the discussion on a concrete example we shall consider a wireless temperature sensor. Nearby nodes may be authorised to request the current temperature, or to register a "watch" that will cause the thermometer to send out a reading when the temperature enters a specific range. We wish to make our thermometer useful in the widest range of environments including environmental monitoring, industrial process control and medicine.

We will therefore consider how we can enable our thermometer to support all the security properties that might be required, including *confidentiality*, *integrity* (and its close relative *authenticity*) and *availability*. Contrary to academic tradition, however, we shall examine them in the opposite order, as this often (and certainly in our case) reflects their actual importance. First, however, we have to mention some of the resource constraints under which such networks operate.

1.2 System Constraints

The three main constraints on Piconet, and on similar systems which support ad-hoc networks of battery operated personal devices, are as follows:

Peanut CPU: the computing power of the processor in the node is typically small, so large computations are slow.

Battery power: the total energy available to the node is a scarce resource. The node likes to go to sleep whenever possible. It is not desirable to use idle time to perform large computations in the background.

High latency: to conserve power, nodes are off most of the time and only turn on their receiver periodically. Communicating with such nodes involves waiting until they next wake up.

The consequence of those constraints is that, while strong symmetric cryptography is feasible, modular arithmetic is difficult and so is strong asymmetric cryptography. Where a peanut node (e.g. the one embedded in a camera) interacts with a more powerful one (e.g. the one embedded in a mobile phone or laptop), one may use techniques such as low exponent RSA, with the protocols designed so that the peanut node sticks to the cheap operations of encryption and verification while avoiding the expensive ones of decryption and signature.

More generally, where there is a trade-off between security and (say) battery life, we may want to let the user control this. For example, if our thermometer is used to drive a wall display in someone's living room that shows the outside temperature, then the owner is unlikely to opt for validated and encrypted communication if this means that he must change the battery every month instead of once a year.

One challenge is to integrate this flexibility in the system without introducing major architectural holes of the sort that would allow the attacker, too, to turn off security at will.

2 Availability

Availability means ensuring that the service offered by the node will be available to its users when expected. In most non-military scenarios, this is the security property of greatest relevance for the user. All else counts little if the device cannot do what it should.

2.1 Radio Jamming

In the traditional threat model–derived from the military–an attacker can deny service to the nodes in a given area by jamming the radio frequencies they use. Traditional defences include spread spectrum and frequency hopping, both of which force the attacker to jam a wider frequency band and thus use more power. We will revisit them briefly in section 5 below. However such concerns are of less relevance to the commercial world, where such attacks are dealt with by complaining to the authorities and having the operator of the jamming station arrested.

The novel and interesting service denial threat is different, and concerns battery exhaustion.

2.2 Battery Exhaustion

A malicious user may interact with a node in an otherwise legitimate way, but for no other purpose than to consume its battery energy. Battery life is the critical parameter for many portable devices, and many techniques are used to maximise it; in Piconet, for example, nodes try to spend most of the time in a sleep mode in which they only listen for radio signals once in a while (the period can be set from a few seconds to several minutes). In this environment, power

exhaustion attacks are a real threat, and are much more powerful than better known denial of service threats such as CPU exhaustion; once the battery runs out the attacker can stop and walk away, leaving the victim disabled. We call this technique the **sleep deprivation torture** attack.

For any public access server, there is necessarily a tension between the contrasting goals of being useful to unknown users and not succumbing to vandals. Whereas some applications can restrict access to known principals, in others (such as web servers and name servers) this is infeasible since the very usefulness of the service comes from its being universally available.

If a server has a primary function (such as sending the outside temperature to the meteorological office every hour) and a distinct auxiliary function (such as sending the current temperature to anyone who requests it) then these functions can be prioritised; a reservation mechanism can ensure that the higher priority use receives a guaranteed share of the resource regardless of the number of requests generated by the lower priority uses. (The highest priority use of all may be battery management: if one can estimate fairly accurately the amount of usable energy remaining, then the service can be monitored and managed provided that the process does not itself consume too much of the resource it is intended to conserve.)

3 Authenticity

3.1 To Whom Can a Principal Talk?

In some applications our thermometer will broadcast its temperature readings, but in general it will only send them to recipients who have been authorised in some way. For example, in a hospital, it might be authorised to send temperature readings to any doctor's palmtop computer or any nursing station. But it might also be required to restrict transmission (e.g. of the temperature of a celebrity) to a particular station or device.

The usual authorisation mechanisms (which turn out to be the least interesting in this case) involve a centralised system administrator. This may be implemented as access control lists (the administrator tells the thermometer who is authorised) or capabilities (the administrator gives some principals a signed certificate which they present to the thermometer when they want a reading). However, the ad-hoc network environment poses a fundamental new problem: the *absence of an online server*.

Interactions with the administrator after the thermometer has been manufactured (or personalised by the institution that owns it) may be expensive or time-consuming, as they may entail establishing a network connection to a central server (perhaps using gossiping via intermediate nodes), or bringing a management device physically close to each affected node. In the particular case of a thermometer, the device might be calibrated every six months, at which time new security state can be loaded; however, rapid changes may be too expensive for a central administrator to make regularly.

It follows that the length of any validity period (whether for certificates or access control lists) will be a trade-off between timeliness and convenience. But relying on expiration dates imposes on the nodes the extra cost of running a secure clock–otherwise the holder of an expired certificate might reset a node's clock to a time within the validity period. As many Piconet nodes would not normally have an onboard clock, the classical approach to authentication is suspect. Thankfully, there is a better way.

3.2 Secure Transient Association

The novel and interesting authentication problem in ad-hoc networks of wireless devices is that of **secure transient association**. If a householder owns a device, say a universal remote control, that lets her control various other devices in her home (such as hi-fi and television components, the heating system, lights, curtains and even the locks and burglar alarm) then she will need to ensure that a new device she buys from the shop will obey her commands, and not her neighbour's. She will want to be assured that a burglar cannot take over the heat sensing floodlight in the garden, or unlock the back door, just by sending it a command from a remote control bought in the same shop.

As well as being *secure* (whatever that means), the association between the controller and the peripheral must also be *transient*. When a householder resells or gives away her television set or hi-fi or fridge, it will have to obey another controller; when her controller breaks down (or she decides to replace it or upgrade its operating system), she must be able to regain control of all the gadgets she already owns.

A central authentication service is possible for expensive consumer durables; most governments run such services for houses and cars. But there is no prospect that this will be extended to all durable consumer goods; the UK government abandoned dog licensing some years ago as uneconomic. In any case, there would be very grave civil liberties objections to the government maintaining lists of all PCs, hi-fis and DVD players in the country; the outcry over the Pentium III processor ID indicates the likely level of political resistance. Even the existing central services stop short of managing keys; the replacement of car keys is left to the motor trade, while house locks are completely uncontrolled. So it is desirable that key management be performed locally: the last thing we want is to impose an expensive and unpopular central solution. Yet it would be nice if we could still provide some means of making a stolen DVD player harder to resell.

Another insight comes from scenarios where we have a pool of identical devices, such as a bowl of disinfectant containing ten thermometers. The doctor does not really care which thermometer she gets when she picks one up, but she does care that the one her palmtop talks to is the same one she is holding and not any other one in the bowl or nearby in the ward.

Many more potential applications of wireless devices require establishing a secure transient association between two principals (typically, but not necessarily, a user and a peripheral). For example, there has been significant interest in the possibility of a police pistol that can only fire when held by the officer to

whom it was issued, who for this purpose might be wearing a very short range radio ring: at present, in the USA, a large number of the firearm injuries sustained by policemen come from stolen police guns. Similar considerations might apply to more substantial weapon systems, such as artillery, that might fall into enemy hands.

3.3 The "Resurrecting Duckling" Security Policy

A metaphor inspired by biology will help us describe the behaviour of a device that properly implements secure transient association.

As Konrad Lorenz beautifully narrates [10], a duckling emerging from its egg will recognise as its mother the first moving object it sees that makes a sound, regardless of what it looks like: this phenomenon is called **imprinting**. Similarly, our device (whose egg is the shrink-wrapped box that encloses it as it comes out of the factory) will recognise as its owner the first entity that sends it a secret key. As soon as this 'ignition key' is received, the device is no longer a newborn and will stay faithful to its owner for the rest of its life. If several entities are present at the device's birth, then the first one that sends it a key becomes the owner: to use another biological metaphor, only the first sperm gets to fertilise the egg.

We can view the hardware of the device as the body, and the software (particularly the state) as the soul. As long as the soul stays in the body, the duckling remains alive and bound to the same mother to which it was imprinted. But this bond is broken by death: thereupon, the soul dissolves and the body returns in its pre-birth state, with the resurrecting duckling ready for another imprinting that will start a new life with another soul. Death is the only event that returns a live device to the pre-birth state in which it will accept an imprinting. We call this process **reverse metempsychosis**. Metempsychosis refers to the transmigration of souls as proposed in a number of religions; our policy is the reverse of this as, rather than a single soul inhabiting a succession of bodies, we have a single body inhabited by a succession of souls[1].

With some devices, death can be designed to follow an identifiable transaction: our medical thermometer can be designed to die (and lose its memory of the previous key and patient) when returned to the bowl of disinfectant. With others, we can arrange a simple timeout, so that the duckling dies of old age. With other devices (and particularly those liable to be stolen) we will arrange that the duckling will only die when so instructed by its mother: thus only the currently authorised user may transfer control of the device. In order to enforce this, some level of tamper resistance will be required: assassinating the duckling without damaging its body should be made suitably difficult and expensive.

In some applications we may need to be able to recover from circumstances in which the legitimate user loses the shared secret (e.g. the password is forgotten

[1] Prior art on this technique includes Larry Niven's science fiction novel *A World Out of Time* (1977) in which convicted criminals have their personalities "wiped" and their bodies recycled.

or the remote control is broken beyond repair). To be able to regain control of the duckling, one should allow for **escrowed seppuku**: someone other than the mother, such as the manufacturer, holds the role of Shōgun with a master password that can command the device to commit suicide.

In other applications, only part of the duckling's soul should perish. In fact, our thermometer will typically be calibrated every six months by the hospital's (or manufacturer's) technician, and the calibration information must not be erased along with the patient data and user key when the device is disinfected, but only when it is plugged into a calibration station. So we may consider the device to be endowed with two souls–the calibration state and the user state–and a rule that the latter may not influence the former. So our resurrecting duckling security policy may be combined with multilevel security concepts (in fact, "multilevel secure souls" are a neat application of the Biba integrity policy model [5]).

3.4 Imprinting

During the imprinting phase, as we said, a shared secret is established between the duckling and the mother. Again, we might think that this is easy to do. If at least one of the two principals involved can perform the expensive public key operations (decrypt and sign), the other device then simply generates a random secret and encrypts it under the public key of the powerful device from which it gets back a signed confirmation.

But many of our nodes lack the ability to do public key, and even if they did it would still not help much. Suppose that a doctor picks up a thermometer and tries to get his palmtop to do a Diffie-Hellman key exchange with it over the air. How can he be sure that the key has been established with the right thermometer? If both devices have screens, then a hash of the key might be displayed and verified manually; but this is bad engineering as it is both tedious and error-prone, and in an environment where we want neither. We are not likely to want to give a screen to every device; after all, sharing peripherals is one of the goals of ad-hoc networking.

In many applications, there will only be one satisfactory solution, and we advocate its use generally as it is effective, cheap and simple: physical contact. When the device is in the pre-birth state, simply touching it with an electrical contact that transfers the bits of a shared secret constitutes the imprinting. No cryptography is involved, since the secret is transmitted in plaintext, and there is no ambiguity about which two entities are involved in the binding.

Note that an imprinted duckling may still *interact* with principals other than its mother–it just cannot be *controlled* by them. In our medical application, we would usually want the thermometer to report the patient's temperature to any device in the ward which asked for it. Only in exceptional circumstances (such as a celebrity patient, or a patient with a socially stigmatised condition) would the patient require encrypted communications to a single doctor's PDA. So should we also have an option of imprinting the device with a cleartext access control list (and perhaps the patient's name), rather than an ignition key?

This brings us back to the issue raised at the end of section 1.2, namely how we might enable a single device to support security mechanisms of differing strength. The solution that we favour is to always bootstrap by establishing a shared secret and to use strong cryptography to download more specific policies into the node. The mother can always send the duckling an access control list or whatever in a message protected by the shared secret. Having a key in place means that the mother can change its mind later; so if the patient is diagnosed HIV positive and requests secure handling of his data from then on, the doctor does not have to kill and reinitialise all the equipment at his bedside. In general, it appears sound policy to delegate from a position of strength.

4 Integrity

So far we have seen that denial of service, the goals of authentication, and the mechanisms for identifying other principals are surprisingly different in an ad-hoc network. Is there any role for the more conventional computer security mechanisms? The answer appears to be a qualified yes when we look at integrity.

Integrity means ensuring that the node has not been maliciously altered. The recipient wants to be sure that the measurements come from the genuine thermometer and not from a node that has been modified to send out incorrect temperature values (maybe so as to disrupt the operation of the recipient's nuclear power plant).

4.1 If You Can't Afford Signatures...

Prudence dictates that a patient's temperature should only be measured by a "known good" thermometer, such as one that passed a calibration inspection within the last six months. So it is natural for calibrators to issue signed dated certificates (though some care must be taken if some of the thermometer's prospective clients do not possess a clock). But the certificate could have been replayed by a middleman. What sort of mechanisms should be implemented to prevent this?

If the thermometer can perform digital signatures and the palmtop can check them, the solution is straightforward: the thermometer's calibration certificate can include the node's public key. Where the thermometer cannot perform public key cryptography, the palmtop will establish a common secret with the thermometer using the techniques of section 3.4 and, having verified its certificate, will be able to accept messages protected by a MAC keyed with the shared secret.

At this point, we depart once more from the conventional wisdom of the computer security community. The obvious objection is that, since neither certificates nor IDs are secret, a false device might be constructed which clones a genuine one; and that the only proper way to use a certificate is to certify a public key whose private key is known only to the device. However, this is tied up closely with the issues of tamper proofness and tamper evidentness. If our

devices are not tamper-proof, then the private key can be read out and installed in a bogus device; but if they meet the much weaker requirement of tamper-evidentness (say with sealed enclosures), a forger will not be able to produce an intact seal on the bogus device. So we will have confidence in a certificate which we receive protected under an ignition key that we shared successfully with a device whose seal was intact. (This is the first example we know of a purely "bearer" certificate: it need not contain a name or even a pseudonym.) We will now discuss this in more detail.

4.2 Tamper Resistance

The doctor's reliance on the "genuine thermometer" certificate assumed that, after the thermometer was last inspected, calibrated and certified, it stayed that way. This assumption may be questioned in many applications, especially as in Piconet not all nodes are well guarded, highly personal accessories such as Java rings [11] over which the owner is expected to keep close and continuous control. On the contrary, many nodes (such as broadcasting sensors) may be commodities that are scattered around and left to their fate. With such a model an attacker may, and sooner or later will, modify or forge a deployed node, possibly redeploying the corrupted node in an unsuspecting environment.

This can in theory be avoided by making the node tamper-proof, but it is much easier to talk about this property than to implement it in practice [1], especially within the cost and form factor constraints of personal consumer electronics devices. Under the circumstances, it is not clear how much extra assurance is given by furnishing our thermometer with the ability to do public key cryptography; such a device can have its private key read out just as a device with a certificate but without a private/public keypair can be forged.

In such environments it may often be more suitable to use physical tamper-evidence mechanisms (such as seals) rather than electronic mechanisms (such as tamper sensing switches that zeroise memory). In this case, one must still design the device so that non-intrusive attacks (such as those based on protocol failure, power analysis and glitch attacks [2]) are not practical; it is also necessary to take into account the time that might pass before a broken seal is noticed, and the likelihood of successful attacks on the sealing mechanism [9].

It must also be realised that the tampering may not be limited to the on-board code and keys: a very effective attack on the unattended thermometer is to simply replace its analogue sensing element with a bad one. This attack highlights that even enclosing the entire processor, memory and backup battery in a high-grade tamper resistant enclosure, with only a ribbon connector to interface with the outside world, would still leave us vulnerable to direct attacks on its "peripherals". Bringing the sensor itself within the tamper resistant enclosure may make manufacturing too expensive (the computing and communication core willl no longer be a modular building block) and may even interfere with the proper working of the sensor. So the transducer may be an Achilles' heel, and it may not be worth spending large sums on tamper-proofing the core if the sensor cannot economically be protected.

When making decisions about what level of tamper-proofness or tamper-evidentness a system needs, it is as well to bear in mind that corrupt nodes can be used in a number of ways. Attacks might be immediate and direct, or alternatively the attacker might field a number of nodes which would accept software upgrades from him as well as from the authorised source.

4.3 Software Upload

For nodes to be useful, there has to be a way to upload software into them, if nothing else during manufacture; in many applications we will also want to do this after deployment. So we will want to prevent opponents from exploiting the upload mechanism, whatever it is, to infiltrate malicious code, and we will want to be able to detect whether a given node is running genuine software or not.

Neither of these goals can be met without assuming that at least some core bootstrap portion of the node escapes tampering. The validity of such an assumption will depend on the circumstances; the expected motivation and ability of the attackers, and the effort spent not just on protecting the node with tamper-resistance mechanisms and seals, but in inspection, audit and other system controls.

5 Confidentiality

We find that we have little to say about confidentiality other than remarking that it is pointless to attempt to protect the secrecy of a communication without first ensuring that one is talking to the right principal. Authenticity is where the real issues are and, once these are solved, protecting confidentiality is simply a matter of encrypting the session using whatever key material is available.

In the event that covert or jam-resistant communications are required, then the key material can be used to initialise spread-spectrum or frequency-hopping communication. Note that, in the absence of shared key material and an accurate time source, such techniques are problematic during the important initial resource discovery phase in which devices try to determine which other nodes are nearby.

6 Conclusions

We examined the main security issues that arise in an ad-hoc wireless network of mobile devices. The design space of this environment is constrained by tight bounds on power budget and CPU cycles, and by the intermittent nature of communication. This combination makes much of the conventional wisdom about authentication, naming and service denial irrelevant; even tamper resistance is not completely straightforward.

There are interesting new attacks, such as the sleep deprivation torture, and limitations on the acceptable primitives for cryptographic protocols. However,

there are also new opportunities opened up by the model of secure transient association, which we believe may become increasingly important in real networking applications.

The contribution of this paper was to spell out the new problems and opportunities, and to offer a new way of thinking about the solution space–the resurrecting duckling security policy model.

Acknowledgements

We thank Alan Jones for suggesting the wireless thermometer, a prototype of which had just been built in the context of Piconet, as a minimal but still meaningful practical example.

References

1. Ross Anderson and Markus Kuhn. Tamper resistance–a cautionary note. In *Proc. 2nd USENIX Workshop on Electronic Commerce*, 1996. 180
2. Ross Anderson and Markus Kuhn. Low cost attacks on tamper resistant devices. In Mark Lomas et al., editor, *Security Protocols, 5th International Workshop Proceedings*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer-Verlag, 1997. 180
3. Infrared Data Association. http://www.irda.org/. 173
4. Frazer Bennett, David Clarke, Joseph B. Evans, Andy Hopper, Alan Jones, and David Leask. Piconet: Embedded mobile networking. *IEEE Personal Communications*, 4(5):8–15, October 1997. 173
5. Kenneth J. Biba. Integrity considerations for secure computer systems. Technical Report MTR-3153, MITRE Corporation, April 1975. 178
6. HomeRF Working Group. http://www.homerf.org/. 173
7. Jaap Haartsen, Mahmoud Naghshineh, Jon Inouye, Olaf J. Joeressen, and Warren Allen. Bluetooth: Visions, goals, and architecture. *ACM Mobile Computing and Communications Review*, 2(4):38–45, October 1998. 173
8. IEEE. IEEE standard for a high performance serial bus. IEEE Standard 1394, 1995. 172
9. Roger G. Johnston and Anthony R.E. Garcia. Vulnerability assessment of security seals. *Journal of Security Administration*, 20(1):15–27, June 1997. 180
10. Konrad Lorenz. *Er redete mit dem Vieh, den Vögeln und den Fischen* (King Solomon's ring). Borotha-Schoeler, Wien, 1949. 177
11. Sun Microsystems. http://java.sun.com/features/1998/03/rings.html. 180
12. Kevin J. Negus, John Waters, Jean Tourrilhes, Chris Romans, Jim Lansford, and Stephen Hui. HomeRF and SWAP: Wireless networking for the connected home. *ACM Mobile Computing and Communications Review*, 2(4):28–37, October 1998. 173
13. Bluetooth SIG. http://www.bluetooth.com/. 173

The Resurrecting Duckling
(Transcript of Discussion)

Frank Stajano[1,2]

[1] AT&T Laboratories, Cambridge
[2] Computer Laboratory, University of Cambridge

Hello, my name is Frank Stajano, I'm affiliated with AT&T Laboratories Cambridge, and I'm doing a Ph.D. at the Computer Lab. Today I am going to talk about the radio thermometer as an example of a wireless gadget that one might want to have talking with other wireless gadgets. I'm a bit of a gadget freak myself and I also have plenty of wire coming out of my pockets. It would be advantageous for some of these things to be able to talk to each other. You have a swarm of devices like this and you might imagine, for example, that I take a picture with this camera – this is not one of these models that have voice annotation but you may have seen there are some that do – and I say, "This was the security protocols workshop in Cambridge". It is stupid for the camera to have to incorporate the microphone and digital audio hardware when this is already in my audio recorder (which has to have it because of its function), so it would be so much better if I could just speak into my audio recorder and have this be transferred into the photograph and not encumber the camera with other things.

Similarly if one had a mobile phone, just about the one gadget I hate, you could use this as your gateway to the world of the Internet. All the things that you wear or interact with at short range (a couple of metres) could then use this to communicate with a wider area, for example, get the train timetable from the web page of `railtrack.co.uk` into your PDA.

When we started proposing this, we were just replacing serial cables with wireless and doing lots of other things which avoid plugging stuff into other stuff and being tangled in a network of things, it is always a problem when you do things like that.

Bob Morris: What is the temperature now?

Reply: The thermometer is the principal actor of my talk, but I don't carry an exemplar of it.

Anyway it's obvious that eavesdropping is going to be a problem with this wireless communication, but what are the other issues? As we are going to see, eavesdropping is not at all the most important one, though this is the one that people think of first. So the thermometer is going to be my example during this talk, and it so happens that we built a thermometer in the context of our Piconet project (a networking technology that does these things I describe). You could heat it up and it would start a fan and things like that, radio controlling the fan to say come on when it's too hot. Because I was working on security one of my colleagues said why don't you think of what the security properties of this thermometer ought to be and so this is what I'm going to do now.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 183–194, 2000.

I will deal with the usual security properties in reverse because this is actually the right order. So instead of confidentiality, integrity, availability, I will look at availability first because this is what really counts and I'm going on to integrity, authenticity, and confidentiality last. Confidentiality is absolutely pointless, as you realise, unless you have authenticity because you may be securing your conversation, but it could be with the wrong person.

So let's start by availability. One obvious thing that can happen to deny service is that they might want to jam the radio channel between the devices involved. That's something that's quite hard to counteract. You can have various radio-level countermeasures, such as frequency hopping, spread spectrum, and so on, to try and maximise the power that the attacker has to put into the attack, but even this requires you to have already agreed with your other party how you are going to hop the frequencies, how you're going to hide the pattern in the spread spectrum, and so on, so it is not something that can be used for the essential initial phase of resource discovery when this thing appears into this environment and then has to find out which other things are near it to find out what other services might be available through it, or it might offer its services to.

So this is a thing that needs to be characterised, but it is quite difficult to do much about it, in the context of these low power, low budget devices. CPU exhaustion is the classical denial of service attack, where something tries to make you do more things than you really want to do and you don't have the resources to do the things you really want to do, but the more interesting attack along this line, which applies to these small portable devices, is battery exhaustion because, unlike your typical Internet server where you may be SYN flooded but you're attached to the mains socket, in this case you have a finite amount of energy contained in the thing and when this runs out then you can't do anything else until the service engineer comes and replaces the batteries.

So this is something that will disable you permanently, that attacker doesn't have to keep on banging at you to deny service if, once it has run down your batteries, you're done. And this is something that you will counteract with similar measures, namely some form of resource management and saying, OK now we only allocate that much of my energy to this job and I will reserve some of this for some other thing I want to do later. The difficulty with this is that you may then use up so much of these resources to do the management itself that it may not be worth it. And then outside things might try to pervert your system software by inserting viruses and things like this, and this is something that affects not just the availability but also the integrity, and we'll talk about it later as well when we see the tamper-resistance requirements of our devices.

I mentioned that the countermeasures had to be cost effective, and when we use cryptography to guard ourselves against some of the attacks, we must be aware of what these small devices can do. The constraints are, that we have a peanut CPU so it can't compute very fast, it is battery powered, so it can't do very much at all, even if it's slow. You can't assume that at night while everybody is sleeping and not asking you to do computations you can pre-compute lots of things that you can use the next day, you can't do that because you run out of

battery. So this means that while the strong symmetric crypto is OK, you have to be very careful about using modular arithmetic stuff because it might take up the order of minutes and it might be very impractical on something which is walking around.

One thing we can do is to exploit asymmetric cryptography as in low exponent RSA we make encryptions much cheaper than decryptions, and verifications much cheaper than signatures. So if we design our protocols around this asymmetry we ensure that the peanut nodes do the easy things but try never to do the expensive ones.

If you look at the problem of authentication in the context of our thermometer example – we have a thermometer which might be in a healthcare situation, for example – and you only want the thermometer to talk to someone authorised to read these things, for example as the doctor that's looking at his patient, and so this Alice must have a way of identifying this other principal Bob, and you have all the usual issues that you would expect. You can have an access control list where, when you make Alice you have in it which Bobs it's free to talk to, and you can endow Bob with a certificate that he can exhibit to Alice and say, OK, I am someone that you can send your temperature readings to. And then you have problems with revocation, and the big difference in this system, which is not one of the usual issues, is that you have no on-line connection to a server that will help you resolve this discussion, because these devices have no guaranteed connectivity with anything, because their radio range may be only, let's say five metres. So there's no way that they're going to be able to contact the manufacturer that put up the top level certificates in them, or any certification authority being delegated by this manufacturer. And so everything has to be resolved with interaction at first between the principals that are available there and then. So if Bob receives a certificate that says, I am someone that your thermometer can send readings to safely, if Bob then misbehaves there's no way that a revocation notice can get to Alice so that she can stop trusting him. So if you could have an expiry date on certificates, you can at least deny reissuing the certificate, that's what you've got to do is time revocation.

Another interesting thing that happens with those systems is that you need a way to secure transient associations between pairs of such devices. So when you take one of these things from the shop, you take it home, open the box, who owns this thing? (New stereo, heating controller, phone, whatever). If all these things are going to be controllable by my Palm Pilot, like a magic wand in my hand, I don't want someone to come to the window and point his Palm Pilot and start running the washing machine or making phone calls on my bill and so on.

So, similar, but different things, who owns the device or tool. This picture of the thermometer example, there's a number of thermometers there in a hospital, and a surgeon takes one and it now has to be that one that sticks to the PDA of the surgeon, and the surgeon doesn't really care which one of these he gets, but he cares that the one that then talks with his PDA is the one that he has in his hand and not another one that makes an association with it. And this thing

applies to a number of other cases, the appliances which we discussed around the house, car radios, pistols (you might want your weapon to only fire when it's under your control so that if someone steals your pistol they can't fire at you with it), and things like this.

So the model I propose for this sort of interaction, what I call the resurrecting duckling and the duckling comes in from this Konrad Lorenz type of thing where when it hatches from the egg the first thing it sees, it's Mummy, so this is the way it would work with these devices. The first thing it sees when it comes out of the box is Mummy, it's the one it would trust. Many of these things in the resurrecting duckling actually happened with some security systems, for example, when you initialise your workstation, if you are the one who starts with the formatted disk and puts on the operating system, you are the one who gets to choose the administrator password, so you are Mummy because it sees you when it starts. And then there's an issue of atomicity, in effect this imprinting situation which happens when the device is born, is in a sense like a deep reset and you want this to also be linked to the fact of seeing you, because otherwise you might reset the device and it might get associated with someone who just pops in front of you, so there needs to be some atomicity between this birth and the imprinting. And so rather than having a button, for example, you might want to have an electric contact into which you plug something and you say, OK the resetting is done by hooking this into that, and as I do this I also transfer a secret between duckling and me, Mummy.

Resurrecting – I had not dealt with this – you know this thing that is associated with Mummy at birth may also die and then resurrect, but this is a reverse resurrection. Metempsychosis is the transmigration of the soul, so when you die you are reincarnated to a higher form of life, or maybe a worse one if you're been bad. But in this case it's the contrary, because the soul dies and the body stays there and it gets a new soul. So instead of the soul being the one that's preserved from one reincarnation to the next, it's the body, the soul only ever lives in one body and then disappears forever. The soul is, to a first approximation, the secret you established when you do the imprinting. So when the soul dies the body returns to the pool like the twenty thermometers. The soul may die for many reasons. It may die because of suicide so it says, OK I am no longer needed now, I can just give up so that the thermometer can be reused, that would be suicide. Old age, you can have a timeout where it says, OK this is my birth, I am now going to be imprinted to you for the next three hours and then I'll just die again and so the thing will return to the pool again. Or it could die because of assassination, so someone comes and steals my laptop and they can't use it because it's imprinted to me, but they reformat the hard disk and put on the operating system so that they can use the hardware. So, you know, the soul never switches allegiance, once it's been imprinted it stays imprinted to the same principal, but if you want to steal the hardware then you need to kill that soul and put in a new one. So if you want to prevent the hardware theft you have to make this assassination more costly than the value of the product.

This is something that does not happen in the case of computers, for example, because it's trivial to reinstall the software on top of it, you lose the soul, as I said. The soul never switches allegiance, you won't be able to find out what the password or the imprinting was, but you can establish a new imprinting with another principal. Whereas if you want to make the stolen laptop inoperative, you have to do something like this, set up the password in the some flash ROM that can only be taken out by proving that you own the original password, and then this would be the suicide, where you say OK, it's me, but now I want to die, or you have to make me take that flash ROM apart, dismantle the whole computer, which would to a first approximation cost you more than it's worth and what you get with just the hardware. This escrowed suicide I also mentioned is something that you have when you have the protection of the type I just described, but then the manufacturer has some sort of escrow route through which he can reset your device. Suppose you are the legitimate owner of this device, you have established an imprint, and you haven't used it for six months and you've totally forgotten how to, you don't know the password anymore, it's one where you paid £1000 for it, so you don't really want to throw it away. So you phone the manufacturer and say, oh how do I unlock my car radio, and they say, please bring me your serial number, and then they make some calculations and they give you a way to kill that soul establishing a new association or something like this. So I found this a model that describes most of the things that actually go on and also the things that you would like to go on in this sort of secure association.

At the act of imprinting you need to establish a common secret, so how do you do that? We said that these devices had peanut processors, if you are talking to another device that is not a peanut but a coconut that can do bigger calculations, then it's very easy because – assuming you can trust the peanut to think up good random numbers – you just have the peanut send the coconut a random number which should be a session key encrypted with the public key of the coconut. That's cheap and so the peanut can do it and leave the hard thing (decrypting it) to the coconut.

Systems that are using many devices that only have symmetric key crypto and are based on key diversification, are not really suitable. This would be something where each device has a serial number and then devices are grouped into categories, let's say thermometers and readers of thermometers, and you would consider the category of readers of thermometers as being somehow master with respect to the slave being the thermometer, and so you encrypt the serial number of the thermometer with the master key for the displays, and this is the secret key that is then shared between this thermometer and the master. In fact it's shared between this thermometer and every principal of the class master of thermometer, and so if you just have one traitor inside the class of thermometer displays then this whole scheme doesn't work.

One very basic but fundamental way to establish a common secret is a physical contact, taking the cable and plugging it into the device and just exchanging a key in this way. This is also desirable for another reason which is, when you

have this bowl of thermometers you can do all the clever protocols in the world but you have to be sure that these are being done with the actual thermometer you have there. If you have a number of devices here radio is not really going to tell whether you are talking to this one, this one, this one, in effect you are broadcasting over all of them, and so you have to have a way of being sure that you are talking to the right one. Physical contact might, in several cases, be a perfect thing to do.

Bruce Christianson: Physical contact doesn't solve the problem of agreeing the sequence, unless you've got some way of knowing the that thing you're physically connected to is a thermometer and not an emulator that I'm trying to use to grab your password.

Reply: Yes I agree with that. If Bob wants to receive readings only from a genuine thermometer, Bob doesn't really care if anyone else overhears the reading of this thermometer, he just wants to have readings from a thermometer that hasn't had its code modified or things like that. One could say Alice can exhibit a certificate from the manufacturer that says this is a real thermometer and so on, and also Alice must be challenged to avoid some maverick having recorded the genuine temperature readings on a cold day and replaced them with a hot day and things like that. So you might want to have Alice send out a signed temperature together with a challenge, but really Alice can't afford to sign each one of these readings like we said, so that's why we would rather do something like this. We first establish a connection between the two devices in some expensive way where we have set up this common secret, and then after this, instead of signing, we just compute a MAC which costs the same as the symmetric key crypto and, for our purposes, is as good as the signature. And the differences between it and a signature don't really matter for what we want to do with it.

Now this is where I come back to you, Bruce. The point is that these devices are subject to being tampered with in several ways. First of all they are in the hands of the user, so the user can open the device he owns and do whatever he likes with it, like a smartcard or a Java ring. But there is also a reverse, so it means that if you are the bank you can't put any bank-like secrets into the thing you distribute to the users. There's also another thing which is that – unlike the smartcard or the Java ring – in the system we are envisaging many of these things will be just scattered around the place like beacons, for example, here is a node that sits there for the purpose of telling the public what the exact time is or what the location is (with GPS), or what the temperature is, or any of these other things which are not actually owned by anyone, they're just there to give out information by radio. So these things, obviously anyone could go there during the night, open it up, change it completely so that it does something completely different, and then put it back and nobody will be the wiser.

Because in our system we allow software to be uploaded into the node, we want to be able to stop Mallory from doing so himself and also spot whether he has done so. And this cannot be done if we don't have some axiomatically tamper-proof core that we know, if this code is in that it's running and it has

not been messed with because if he can do anything to the node then he can just substitute with something else inside the same case. But even this is not the whole story, even if we had something, we axiomatically assume that something like an i-button is secure, it still isn't good enough. Because this is, for example, the thing that IBM make for secure crypto processing in real life, it's about this size encasing all sorts of clever things, in fact whatever you might do with it, but then it has a ribbon cable coming out to its peripherals. Now if this is our secure thermometer, after this ribbon cable there would be an analog sensor of the temperature, and if I can substitute with one that gives the wrong temperature then all the crypto is basically useless, because I can't protect with crypto the analog link between the sensor and the A-to-D convertor.

Virgil Gligor: Does it have an internal battery, so you can have a thermometer inside it?

Reply: Yes, it does, so you can have a thermometer inside it but then it might not have the same sensing properties. I am making the point that although I need something that is tamper-proof to run the software that I will use – for example, to upload more applications into it – even having the most robust, tamper-proof processor and memory and battery, whatever, is not going to be good enough, because I need to protect the total integrity of the node *including its sensors* otherwise I have a way of tampering it that would be undetectable with crypto.

Virgil Gligor: Yes, there was a bunch of research done in the eighties about how to build tamper-proof boxes that emphasised that if you have independent power you could have voltage, pressure, temperature-controlled, self-protecting devices and I was wondering if it did more than that here.

Ross Anderson: I think there is a form factor issue here, in that a 4758, being 2cm × 8cm × 16cm is a little bit large for taking a rectal temperature.

Bruce Christianson: If it's taking an EEG rather than temperature, it's too small, you can't fit the whole thing inside.

Reply: Yes, another point is that their enclosing the whole thing in a tamper-proof enclosure is in fact way out of budget for the sort of target that we are aiming for.

Virgil Gligor: Precisely, what does tamper-proof mean here?

Reply: What it really means is that it will set a cost limit on what you can expect before it's broken. Once you know this then you know how much the secret's going to be worth inside it and you say, OK it's pointless trying to use this for things that above a certain level, because you can break it anyway just by opening it up for less than that.

Virgil Gligor: I think that's the right answer. I talked to some of the people on tamper-proof PCs, and they showed me how one could construct a tamper-proof PC which had been tamper-proof for 15 minutes given normal tools, tamper-proof for two hours given normal tools, tamper-proof for five minutes given abnormal tools, so it's basically a measure of time and attackers tools.

Reply: I agree. So given this, one should think of making the physical seals also part of the security architecture. But this also doesn't get you home free,

because if you start relying on physical seals which are something outside the protocols then something that is inanimate like a thermometer will not be able to verify me. So it's OK if I, doctor, look at the thermometer and see if it's been tampered with or not, but the thermometer itself can't see if I'm really a doctor or if I'm a crook pretending to be a doctor.

Michael Roe: That's related to a thing we see with smartcards. As has been shown by a great deal of research, most smartcards can be tampered with, given an attacker with acid baths and lasers and all these other strange pieces of equipment. And the usual answer is, that you design your protocols so that the person who has physical ownership of the card is not the one who has the incentive to prise the secret out. And then you're OK because you just don't let strangers take your card away from you and dip it in an acid bath and all these other crazy ideas. I think you've got a similar problem here, so you construct your protocols for using the device so that the person who's physically holding it is not the one who has the incentive to make it go wrong.

Bruce Christianson: You ensure that the process of tampering leaves some visible trace and you ensure that some party, who has an incentive to be honest about it, inspects the device prior to the protocol conclusion. So you might have an incentive to break into your own smartcard, but if the protocol says I get to look at it and see that the seal is intact, then all the money's safe.

Virgil Gligor: This principle has been used by a number of people who cared about security: having one-time adhesive tape with a registered serial number on it. So you seal your PC with adhesive tape in the right places and you can tell if somebody tampered with it – reasonably effective for some technologies.

Reply: As I said, in the specific environment that I'm researching it must be realised that this then leaves out some cases which would be interesting. In cases of do-nothing technology, where you take advantage of the wireless by saying that every time this digital camera is on my desk then it saves all these pictures on the hard disk, and I don't actually do anything, I just go near my desk from time to time and it just happens. Nobody's going to check this, because the beauty of this is that I don't do anything, so I am not going to see whether it has the sealed tape with the serial number. So this doesn't protect from these sort of things which are central to the novelty of my research.

Other things that I won't go into but are certainly relevant, are a more systematic and quantitative threat analysis. I have presented some of the things that can go wrong, but to make a secure architecture for an environment like Beacon, one would want to analyse more systematically all the possible things that could go wrong and assign probabilities and costs. Costs both for how much does it cost me if it goes wrong, and how much does it cost him to make it go wrong, and so on, how much does he gain from it going wrong, but this is hard to do without referring to specific applications for the wireless nodes. I find it hard to just think of it as four nodes without thinking of the thermometer or the camera, because then it would be different environments and different incentives for the various entities.

And then of course there's the issue of delegation. I don't think of this system working with just the manufacturer granting certificates, some of the certificates will have to be granted by someone else like a maintenance engineer, or just the owner of the device, and so on.

So I'll open the floor to discussion now. This is just a summary of what we've seen, this environment has new types of denial of service attacks, the key thing is there is no on-line server to rely on because the connectivity is intermittent and localised. So it's useful to have things that rely on public key but you then can't revoke them. We have a small processor that can't do much computation and can't do it very fast either, so we have to optimise the protocols so that we don't do the things it can't do, very obviously. And we have issue of a secure transient association for ownership and control of devices, I have presented the ressurecting duckling model to explain how this all fits in. And then these are things that are bound to be subject to tampering and so you have to take into account the seals, so that you see if the thing has been messed with, but this spoils the purity. I wrote that just as a reminder of the fact that then you can't do things without inspecting the seals, which is something the device is not able to do by itself.

So there are some new constraints and some new goal functions with respect to these old well known properties that we have examined in other contexts.

Virgil Gligor: One thing that you should really think about in proposing security in this framework is that of administering. This should be a zero administration effort system, in other words, perhaps when you purchase a particular device and you throw this into all the others in your personal network, there may be some administration but none beyond that, or close to zero, because if any administration is required on the part of the user, you will sell absolutely none of these devices. So have you thought about the administrative aspects, what do I have to do other than just having these devices in my possession?

Reply: Yes, this is a very good point because when thinking about the delegation aspects that I haven't mentioned here, just hinted at the fact that they existed, then who sets the policy for what other devices this thermometer can talk to, etc, etc. You have a tension between wanting to be able to give the buyer of the thermometer the liberty to choose the right policy and wanting to have something sensible there so that if you just buy it and use it you will do something right instead of just paying.

Virgil Gligor: I mean, a very simple example, how many times did you change your access controls on your own files, I haven't in the last six months, and I just set the `umask`. I mean this is the kind of behaviour you should expect from normal users, and even that's too much for the kind of security that you are talking about.

Reply: Yes, I agree.

Bob Morris: I'm not so sure I can agree, every time he's used the word thermometer I have assumed he really meant thermonuclear warheads, that's the kind of guy that I am, and now I think you have to rather rethink and reword your notions of budgets on both sides.

Virgil Gligor: See what context does to an honest person! I want to sell these devices, Bob, I want to make money, I don't want to lose money.

Matt Blaze: I think it's interesting that in the example you showed us, even though it's kind of a toy example, a thermometer, it occurred to me that the security services you provide, although they might seem kind of frivolous in the example that you gave, in fact also provide you with services that aren't traditionally considered security. In particular if I'm using a remotely controlled thermometer, I really want to be sure that these readings came from this thermometer, not because of some malicious threat, but because I would like that assurance so that I can rely on the data, and by providing a security infrastructure, you're providing this other assurance for free.

Reply: Yes, why do you not consider this as security?

Matt Blaze: Well, only because the threat model is not malicious. I'm not worried about evil thermometers interfering, I'm just worried about whether this reading is in fact faithful and accurate and if I've got strong authentication between the devices and the link has been authenticated and so on, so I'm sure that no spurious readings are getting there, I get that service.

Reply: I see this on the same league as reordering the confidentiality, integrity and availability triad, because I think that seeing things like availability and authentication and so on as important security properties, is the right way.

Virgil Gligor: I think there is also a secondary point here which may be quite important, namely, you've got all those devices networked in some way, therefore you build a dependency between those devices. I think the thermometer right now is connected to absolutely nothing so I tend to trust that if anything goes wrong I see it in the thermometer, it went wrong in here, it didn't depend on anyone else. When you network all these devices together, all of a sudden operation of one of my devices, which used to be stand alone before, depends on somebody else. So the problem that may occur is that errors, simply failures, start propagating around, which they didn't before because they had isolated devices. So it's not just the security of the device in the normal sense and the denial of service in the normal sense, but now the failures of one device can actually affect all the others, which didn't happen before when they were stand alone.

Reply: In any network environment, if you rely on other services then you are vulnerable to their failure.

Virgil Gligor: That's right, so we are really talking about a different form of personal devices here. I have some personal devices which I may not want networked ever. They are so personal I don't want them networked.

Reply: Because you don't trust what would happen otherwise?

Virgil Gligor: I rely on them too much. So maybe there is a class of devices that I don't care that much about and they are useful in some way, and I network them together, perhaps there is a way to distinguish between these two classes of devices.

Michael Roe: Medical devices like heart pacemakers, things like you're going to die if it stops working, you don't want depending on a key distribution service and a bunch of other stuff.

Reply: That's really more of a reliability concern.

Matt Blaze: I think I'd rather have a networked pacemaker than one in which I have to undergo surgery in order to have it reconfigured.

Reply: Possibly.

Bruce Christianson: But Matt's point is that a good security infrastructure along these lines actually is a good way of leveraging a lot of good things like reliability and protection against administrative incompetence and so forth. If you've got the right security infrastructure you can get all these other things ...

David Wheeler: I was going to ask you on a side issue, you mentioned jamming and presumably your thermometer will increase its signal level when the surroundings are noisy or being jammed, so it will use more battery power, but owing to the difference of bandwidth between the jamming and your signal, you should actually rectify the jamming power and use it to drive your signal above the jamming level, so it can automatically defeat it.

Bruce Christianson: There's one other point I was going to make which is, you've put what you're doing in the context of devices that are going to stay within a few metres of each other, but this idea of saying we have a bootstrap which involves physical connection following which we have a more nebulous way, there's a lot of places where it's useful. For example, where I want to put a token in Mike's domain and I'm going to interact with that token in some way, and there's a rule that says if Mike can't produce a token with a tamper-proof seal intact then any doubt is resolved against him. So he has an incentive to keep the token safe, but we need to establish provenance, we need to establish that it's the right token and it's using the right key. A mechanism like this, where you can plug the tokens together, run it through a protocol, separate them and then go to another protocol at a long distance, you know, over the Internet or over the phone or something like that, there's a lot of applications for this type of work in that area as well. So I think there are some weaker assumptions under which what you're saying is very interesting.

Ross Anderson: It just also occurred to me, when Virgil was saying the most critical things he didn't want networked, that perhaps the most critical device you can use many times a day as far as your physical survival is concerned are the brakes in your automobile. Now whether you like it nor not, they're being networked. They're being networked with systems like CANBUS, and so the ABS is running off the same LAN, for want of a better word, as all the rest of the car's electronics and – if it's a heavy goods vehicle – the tachograph, the speed limiter and all that stuff. Now the interesting thing about this is that we have been looking in the context of work with the Ministry of Transport at how you going about securing the pulses between gearbox sensor and the tachograph in the lorry, so people can't fiddle their tachographs or their speed limiters, and the critical problem is key distribution. I had to spend some time explaining to one of Britain's largest truck component manufacturers that it's not a good idea

to put the same DES key in all 200,000 gearbox sensors that they manufacture every year, and it just occurred to me that this kind of resurrecting duckling approach is maybe the right way to do key management, because you let the gearbox sensor send a key to the first vehicle unit that he sees, and if you want him to commit suicide and send another key you'd have to crawl right under the truck with your screwdriver and do something. That at least would stop the attack scenario where the bad guy has got two vehicle sensors, one of which is sitting in his gearbox and the other of which is sitting under his driver's seat and he switches between them so he can take rest periods while he's driving.

Virgil Gligor: I agree with you that you occasionally network very sensitive devices as part of a system. On the other hand in all areas where that's done, devices such as brakes or fly-by-wire devices in an aircraft, the reliability aspects of that system are at an equal level of concern and probably much higher than security aspects, which means that the security threats that we're talking about are considered not to be at the same level as the others and you have some new way of designing security.

Ross Anderson: Security is the main concern with tachographs.

Reply: Maybe there should be a distinction between networking something for the purpose of output of information and for the purpose of control. There should be a main distinction. If you're concerned with reliability you don't want any incoming control, you don't mind if you have outgoing sensor data.

Virgil Gligor: Yes, but a larger point, I think, was made by somebody else – that once you start doing this, the distinctions between security and reliability and availability become rather blurred and the end user couldn't care less what problem went wrong. So you have to look at all of them, not just one. You have to look at the entire system.

Michael Roe: Increasing security can sometimes make reliability worse if, in order to do all this authentication, you build in dependancies on infrastructure like key distribution services, so that you can build yourself a reliability problem.

Reply: And also if you have to say, you can't do that because this security check hasn't gone through.

INTERNET-MARKs: Clear, Secure, and Portable Visual Marks for the Cyber Worlds

Hiroshi Yoshiura, Seiichi Susaki, Yasuhiko Nagai, Tsukasa Saitoh, Hisashi Toyoshima, Ryoichi Sasaki, and Satoru Tezuka

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

Abstract. Visual marks play critical roles in the physical world, but their use in the cyber world is limited because there they are easy to forge, tamper with and copy onto unauthorized data. This paper therefore describes a new type of visual marks that are secure because digital signatures are embedded in them and that can be used with a wide variety of cyber-world systems. It also shows the effectiveness of these marks by describing their application to WWW site authentication.

1 Introduction

Visual marks are widely used in the physical world, where they are familiar to nonprofessional people and can convey information briefly and clearly. As the cyber world becomes more and more like the physical world – filled with money, mail, and shops used by nonprofessional people – there is an increasing need for visual marks in the cyber world. The simple incorporation of visual marks, however, causes serious problems because in the cyber world these marks are easy to forge, tamper with, and copy onto unauthorized data. Visual marks suitable for use in the cyber world must be clear (i.e., easily understood) and must be secure (not easily forged, tampered with, or copied onto unauthorized data). They must also be portable, or easy to use on the wide variety of continuously evolving systems making up the cyber world. This paper describes a new type of visual marks meeting these requirements.

Sect. 2 proposes the new visual marks named INTERNET-MARKs, Sect. 3 describes an application of INTERNET-MARKs to WWW site authentication. Sect. 4 compares INTERNET-MARKs with alternative approaches, and Sect. 5 concludes the paper.

2 INTERNET MARKs

INTERNET-MARKs are made of drawings and are simply image data, such as bitmap graphics or JPEG files. They are placed on data that represent objects in the cyber world, and they carry information about the data. Actual operations of "placing I-MARKs" may be pasting the corresponding bitmap or JPEG on the data.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 195–202, 2000.

Fig. 1. Examples of FIGUREs

2.1 Basic Structure

We define some terminology.

FIGUREs drawings from which INTERNET-MARKs are made.
DATA data on which I-MARKs are placed.

Fig. 1 shows examples of FIGUREs. We also abbreviate INTERNET-MARKs and digital signatures as I-MARKs and signatures respectively. As shown in Fig. 2, an I-MARK is a simply FIGURE into which a signature is embedded by digital watermarking [1]. This signature is a signature for both the FIGURE and the DATA on which the I-MARK is pasted. Additional application-specific information may also be embedded into the FIGURE.

2.2 Security Systems

An I-MARK is issued (Fig. 2) by having the issuer sign for the DATA and the FIGURE and embedding the signature in the FIGURE. And as shown in Fig. 3, an I-MARK is verified by first cutting it out of the DATA and then extracting and verifying the signature. If the verification succeeds the system guarantees the following:

- The I-MARK is not forged and has not been tampered with.
- The I-MARK is on authorized data.
- The DATA has not been tampered with.
- The I-MARK was generated and placed on the DATA by the person indicated by the verification key (public key).

2.3 Properties

(1) Clarity
I-MARKs are easily understood because the watermarking does not degrade the clarity of the FIGUREs from which they are made.

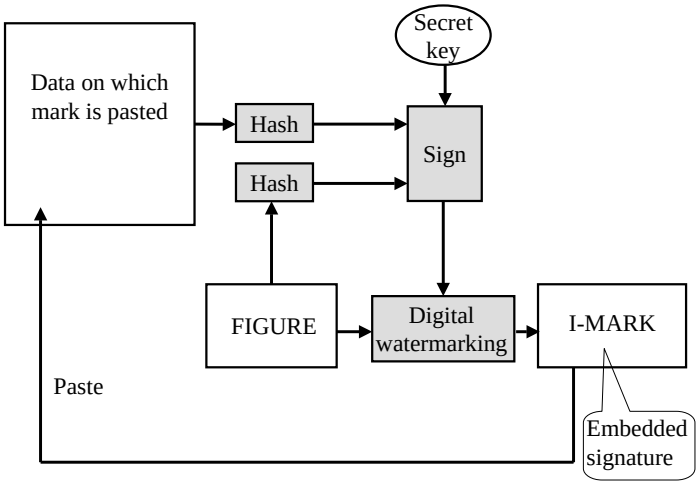

Fig. 2. INTERNET-MARK issuing system.White and black represent data and processes respectively

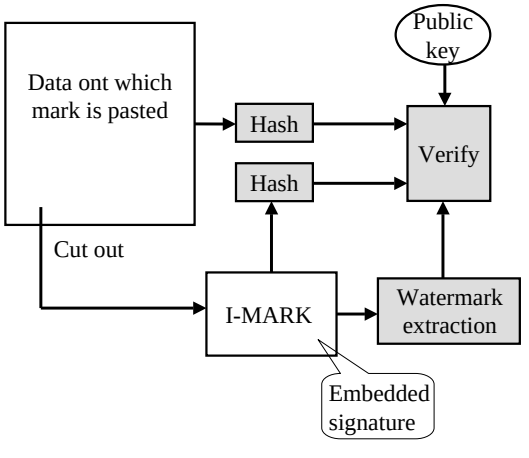

Fig. 3. INTERNET-MARK verifying system

(2) Security
It is obvious from Fig. 2 and Fig. 3 that the security of I-MARKs is equivalent to that of their underlying signatures.
(3) Portability
Equivalent clarity and security could be provided by using a simple combination of visual marks and digital signatures, but I-MARKs are more portable than the simple combination because the clarity measures (visual figures) and the security measures (digital signatures) are amalgamated in a single object.

3 An Application of INTERNET MARKs

This chapter illustrates the effectiveness of I-MARKs by describing their application to a WWW guaranteeing problem. There are three players in this application problem:

- WWW site owner (abbreviated as Owner).
- WWW site user who accesses the Owner's WWW site (User).
- Person who issues a guarantee for the Owner's WWW site (Guarantor).

When the owner asks the guarantor to issue a guarantee for owner's WWW site, the guarantor issues it and sends it to the owner, who places it on the pages of the WWW site. The guarantee may be a rating of the site, a certification of its suitability for use in schools, or any other information relevant to the site. The user accessing the site can get information about it simply by looking at the guarantee and can verify the guarantee when necessary (e.g., when sending a credit card number to the site).

We use I-MARKs for guarantees, and Fig. 4 shows that the system for issuing these guarantee I-MARKs is simply the basic I-MARK issuing system extended to include a signature for IP address of the WWW site to be guaranteed. This signature is needed to prevent WWW site disguise: the copying of both the WWW page data and its guarantee to the WWW site of an attacker who pretends to be the legal owner of its contents.

3.1 Protocols of the Application System

We first define some terminology.

SK_x	secret key of player x.
PK_x	public key of player x.
Enc(DATA,K)	result of encrypting DATA with key K.
IP-ADRESS	IP address of the WWW site to be I-MARKed.
W-DATA	DATA defining pages of the WWW site. This may be HTML source codes.
X | Y	concatenation of data X and Y.

The protocols for issuing and authenticating I-MARKs are illustrated in Fig. 5, and there are four steps in the protocol for issuing them.

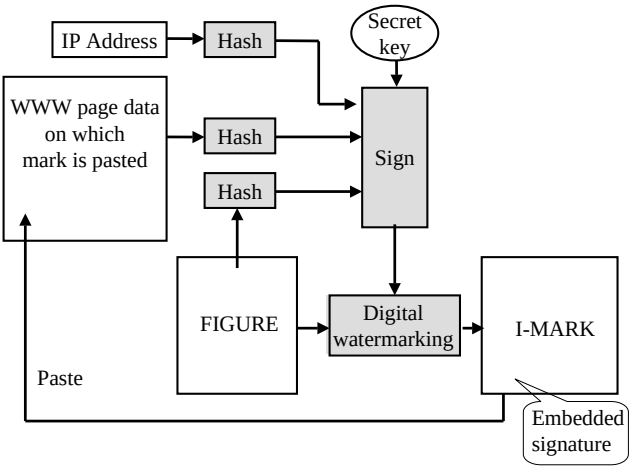

Fig. 4. System issuing an INTERNET-MARK for WWW authentication

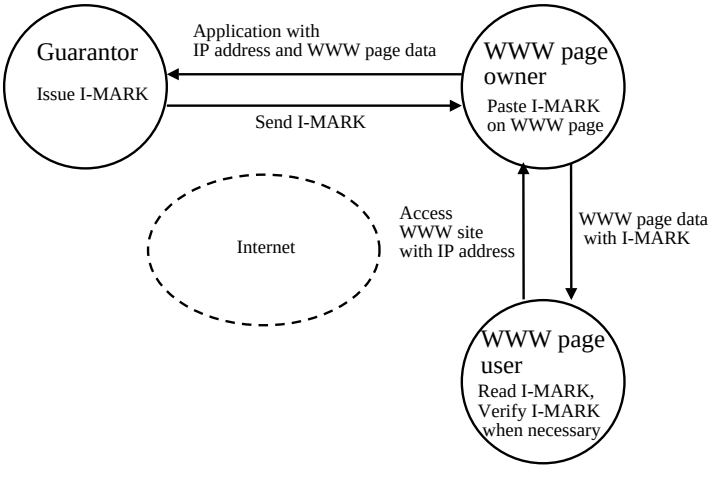

Fig. 5. Protocols for INTERNET-MARK application

Step1 The owner applies for an I-MARK by sending Enc(IP-ADRESS | W-DATA | Memo,SK_{Owner}) to the guarantor. Memo specifies the kind of I-MARK the owner wants.
Step2 The guarantor issues an I-MARK by the following substeps:
(2 - 1) Obtain the IP-ADDRESS,W-DATA,and Memo by using PK_{Owner} to decrypt the application.
(2 - 2) Select a FIGURE appropriate to the Memo and then generate a signature for the FIGURE, IP-ADRESS, and W-DATA. This signature is Enc(Hash(FIGURE) | Hash(IP-ADRESS) | Hash(W-DATA), $SK_{Guarantor}$).
(2 - 3) Generate an I-MARK by embedding the signature in the FIGURE.
Step3 The guarantor sends Enc(I-MARK,$SK_{Guarantor}$) to the owner.
Step4 The owner pastes the I-MARK on the WWW page.

The protocol for authenticating I-MARKs has 3 steps.

Step1 A user accesses the owner's WWW site by using an IP address.
Step2 The owner sends WWW data (i.e., HTML source codes) accompanied with an I-MARK to the user.
Step3 The user gets information by looking at the I-MARK and when necessary verifies the I-MARK by the following substeps:
(3 - 1) Cut the I-MARK from the WWW data, and extract the signature from the I-MARK.
(3 - 2) Verify the signature by using $PK_{Guarantor}$.In the verification, hash values are calculated for the HTML source codes and IP address of the accessed site and for the FIGURE that was used to make the I-MARK.
(3 - 3) If the verification succeeds, the protocol guarantees the followings:
– The accessed WWW site is the one for which the guarantor issued an I-MARK.
– The I-MARK is on the intended W-DATA.
– Neither the I-MARK nor the W-DATA have been tampered with.

3.2 Properties

These application protocols have been implemented in C language, and Fig. 6 shows I-MARKs made from the FIGUREs in Fig. 1. A roughly 2-K bit digital signature is embedded in each I-MARK.

(1) Clarity
As can be seen by comparing Fig. 1 and Fig. 6, the clarity of the FIGUREs is not degraded by watermarking.
(2) Security
Four types of attack are possible.
Type1 Attack I-MARKs themselves, i.e., forging and tampering with I-MARKs.
Type2 Copy I-MARKs onto unauthorized data.

Fig. 6. INTERNET-MARKs generated from FIGUREs in Fig. 1

Type3 Copy both a WWW page data and its I-MARK to the WWW site of an attacker who pretends to be the legal owner of the WWW data.

Type4 Intervene in communication between the owner and guarantor to tamper with the owner's application for an I-MARK and with the guarantor's returning the I-MARK.

As mentioned in previous chapter, the security against first and second types of attacks is equivalent to that of their underlying signature. The security against third type is also the same as that of the signature because I-MARKs contain the signature for IP address of the correct WWW site. The security against fourth type is guaranteed by the public-key cryptosystem used in Step 1 and Step 3 of issuing I-MARKs.

(3) Portability
This issue will be discussed in next chapter by comparing I-MARKs with alternative approaches.

4 Comparison with Alternative Approaches

This chapter compares I-MARKs with the following three alternative approaches to the WWW guarantee problem.

- Simple Marks: The guarantor issues a mark that the owner attaches to the WWW page.
- Simple Signatures: The guarantor issues a signature for WWW page, and the owner attaches it to the page.
- Simple Combination of Marks and Signatures: The guarantor issues a mark and signature for both the WWW page and the mark. The owner attaches the mark and signature to the page.

It is clear from the comparison results summarize in Table 1 that simple marks are clear and portable but are of course not secure. And because a digital signature expresses nothing by itself, it tells a WWW user nothing about the

WWW page unless the user goes through the verification process. Simple signatures, although secure and reasonably portable, thus do not meet a user's needs because their meaning are not clear.

A WWW owner using a combination of marks and signatures needs to attach a mark and the corresponding signature to the WWW page in a way that there is a link between them (e.g., clicking the mark causes the signature to be verified). This attachment needs to be standardized so that WWW users can use a common program to verity signatures. Such standardization requires extensive and continuous efforts because the languages for describing WWW pages are continuously evolving as are the WWW managing systems. I-MARKs also require standardization, but this standardization should be much easier because only the way of attaching I-MARKs needs to be standardized. That is, there is no need for standardization of the ways of attaching signatures and of linking marks and signatures.

Table 1. Comparison with Alternative Approaches

Methods	Security	Understandability	Portability
Simple marks	NG	Good	Good
Simple signatures	Good	NG	Not so bad
Simple combination of marks and signatures	Good	Good	NG
INTERNET-MARKs	Good	Good	Not so bad

5 Conclusion

This paper proposed a new type of visual marks suitable for use in the cyber world. The paper has shown that INTERNET-MARKs are as easily understood as conventional visual marks, are as secure as digital signatures, and are more portability than a simple combination of visual marks and digital signatures.

Acknowledgements

This research is supported by the Ministry of Posts and Telecommunications of Japan and the Telecommunications Advancement Organization of Japan.

References

1. M Swanson, M Kobayashi, and A Tewfik,: Multimedia Data-Embedding and Watermarking Technologies. Proceedings of the IEEE, Vol. 86, No.6, 1998, pp.1064–1087. 196

INTERNET-MARKs
(Transcript of Discussion)

Ryoichi Sasaki

Systems Development Laboratory, Hitachi Ltd.

I am Sasaki from Hitachi Systems Development Laboratory. Today I speak about visual marks, secure and portable visual marks for the cyber world. This slide shows the conduct of this presentation. First of all I will show the requirement for cyberworld marks. Visual marks – for example graphic signals – are useful in the real world, it is possible to compare information briefly, and they are clear and familiar to a non-professional. Visual marks are also useful in the cyberworld, but insecure in cyberworld because easy to fool and tamper with, and easy to repeat the marks on unauthorised data. So useful and secure marks are desired for the cyberworld, with proposal – namely Internet-marks – in this presentation.

Before explaining those Internet-marks I will show the function of marks. This part shows the contents of Internet and this part shows the marks. A mark is best on content and it shows comment of mark issuer, for example, this JWA could be WWW contents suitable for all science education. These are examples of marks in the Internet homepage. This mark shows a family friendly hour. There are three requirements for cyberworld marks. First is security, it must prevent fooling and tampering with marks themselves, and tampering with marked contents and moving marks onto unauthorised contents. Second is clarity, it must convey information as clearly as a physical mark. Third is portability, it must be easy to use from existing systems.

Next I propose Internet-marks. First of all I will describe the basic method of Internet-mark provision. An Internet-mark is digital data. A digital signature is embedded by using digital watermarking technology. This is the material corresponding to the Internet-mark and this is normal data, for example data for bitnet or Internet, something like that. The digital signature is a signature for the material content on which the Internet-mark is based.

This slide shows the system for issuing Internet-marks. This is the material you made, and this the content of each mark. Hash values for the material we made and the contents are calculated, and these hash values are used to generate these digital signatures by using a secret key to make sure, and the Internet-mark is generated by embedding the digital signature by using watermarking. After that this Internet-mark is pasted to the contents.

Material is marked for details like, this material is family friendly or something like that. This slide shows the system for verifying Internet-marks. This is the content on which a mark is pasted. First we cut out the Internet-mark from the contents, after that the embedded signature will be constructed from this Internet-mark, after that the process for verifying is the same as that of ordinary digital signatures.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 203–207, 2000.

Next I will explain the applications of WWW authentication. Content of the WWW has become a serious problem, for example, there are many anti-social contents, for example, pornography, violence, something like that. And here is a WWW site which pretends to be another WWW site, something which has been in favour previously. Firstly, PICS. PICS means Protocol for Internet Contents Services. PICS is useful for anti-social contents, but not practical for illegal WWW sites, because a WWW site-owner is not willing to report their own site problematic. Next is a mark showing WWW owner's or the third party's certification, this mark has major problems of security. So the solution by Internet-marks is an improvement of conventional marks, with improved aspects of the security.

This is a system for issuing Internet-marks for WWW certification, and this system is a simple extension of basic Internet-mark system. The signature is generated not only for the contents and material they made, but also for the URL and IP-address of the WWW site. A signature for URL and IP-address is used for detecting WWW misuse, that is the contents and the pasted page of Internet-marks copied onto another WWW site: the other site owner copied the contents and the Internet-marks and then they do the business instead of the original site owner.

This sets out the system for verifying the Internet-marks for WWW authentication. This is also a simple extension of the basic system, the extension of the signature for URL and IP-address is verified to detect that the WWW site is disguised.

Now I will talk about validation. First is the security. Internet-marks make it possible to detect marks for a company, and to detect companies that mark contents and then move marks onto unauthorised content, and detect moving some mark and the contents to an unauthorised site. Next is the clarity. About two kilobits of digital signature must be embedded to the Internet-marks. Judging from our experience of watermarking the clarity is not degraded by the embedded 2Kbit digital signature, as shown here. Third is the portability. A simple combination of marks and digital signature could provide equivalent security and clarity, but a simple combination requires us to specify ways of attaching marks to the contents, attaching signatures to contents, and linking marks and signatures. On the other hand, Internet-marks require us to specify only a way of attaching Internet-marks to content.

Now, I will show the comparison with alternatives. This table summarises the comparison. Simple marks, of course, lack security. A digital signature makes bit strings for users. They express nothing about a WWW site, so users get no information until they go through a verification process, a detailed signature therefore lacks clear meaning. A combination of mark and signature requires to specify the link between them as well as methods of attaching marks and signatures, so the combination lacks portability. In summary, Internet-marks are the best among these solutions, although the difference is small.

Conclusion, we propose that Internet-marks are visual marks which are safe for use in the cyberworld. As secure as digital signatures, as clear as conven-

tional visual marks, and more portable than the simple combination of marks and signatures. This research is sponsored by the Ministry of Post and Telecommunications in Japan, and from July next year a small trial will be performed in Japan. Thank you very much for your attention.

Frank Stajano: You mention digital signatures as lacking clarity for the user because the user has to go through a verification process, whereas in your system you have something like a little sticker that the user immediately relates to. But when the user relates to this sticker, he has no guarantee that this not been taken from another source or something, so unless the user actually does a verification himself then that isn't going to be any better than the digital signature. In fact it's going to be more misleading.

Michael Roe: This is assuming you have got a watermarking method that works. So if you've got the infrastructure designed for copy protection, that stops someone taking an image that you own and sticking it on their own documents. If you've got the mechanism that does that by having something in the browser that can detect the watermarking in the image and says, no that shouldn't be here, it's copied, then you can use this mechanism to make the user interface onto the authentication. Because you can then say, by using this mechanism, that that image that the user recognises can only be stuck on documents that actually come from the source.

Frank Stajano: I have two objections to that. The first one is that if the watermark does it's job then the user can't distinguish between one thing that does have the watermark and another thing that doesn't have it. It's good enough for the forger to exhibit something similar to that mountain in the blue sky, so that the user perceives this as being the mark. The second objection, perhaps more substantial, is that you are postulating that the browser does all these things, that is displaying the mark only if the thing checks out and so on, and that's fine, but then in this case you lose all the argument of portability because then you can't use this as just an image source and very good thing, you have to have a browser based on it that only displays the thing if it has been verified. Is that right?

Francesco Bergadano: Yes, if you were to put the signature in an invisible part of the page, there's many ways to do that, in a zero-sized frame, as an HTML comment, etc, then it will do the same. Why do you have to hide the signature in a complicated way by watermarking rather than just attach the signature to the page somehow, since anyways you get the browser to check it.

Reply: I mentioned earlier that it is possible to use a combination of marks and signature, but in that case we must specify many ways how to connect while attaching to contents and something like that. So we must decide many standards. If we use Internet-marks, then only one specification is required, for attaching Internet-marks, and the computational process is needed anyway.

Bruce Christianson: Also if you render the mark and the signature separately, then a content provider can cut and paste one easily without taking the other with it, whereas this way it's harder for them to do that.

Dieter Gollman: The argument is, you have a browser that looks at material marks in such a way as to display only those images that are tolerated on the local policy as implemented in the browser. So, you can put a signature anywhere you want, but if you put it in a separate frame, or somewhere else, you have redesign all of your control systems, so you put in a place where you're already looking.

Markus Kuhn: I think for most applications I can think of, this is rightly solving the wrong problem, because you are assigning the precise bit by bit content of the URL and the web page. If I'm a school authority and I want to certify that the page of this company is useful content for educational purposes, then actually what I want to certify is that the people who make these pages are qualified to provide this type of content and they should be free to make later updates to make spelling error fixes and things like this. But I think it would be nicer to have some sort of version control system, where you keep track of the history of your documents. And you sign a certain point in the history, this version of last December was in my opinion of the top five percent, and from then on I can see it, OK, you signed the old version, this is the new version, I have a personal browser, I click on a button, I can see the differences that have been edited in the meantime marked in a different colour.

Bruce Christianson: That's sometimes true, but sometimes what I want to say is, this is a very dubious web site but these exact four pages are OK.

Markus Kuhn: The model I think of is a bit more related to scientific journal publishing. I first write a draft, the draft is scrutinised by the referees, then I get the draft back, and then I can basically do what I want, and this would never be seen again by a referee before it gets to the printer. So the referee has trusted that I will not mess up between the draft and the final published version, and I think I can sum up by saying something like this would be desirable.

Matt Blaze: I think when you speak of watermarking you should be careful and make the point clear, because I suspect people have misunderstood. We think of watermarking as a thing that's difficult and doesn't work if someone tries hard to erase it. Of course here it's the opposite situation, nobody wants to erase the signature, and you can use the most fragile watermarking method possible for the watermarking. And then we can see that there's no problem embedding this in the image in the most naive possible way, and that would be simpler than having a field on a page, probably one could even make a plug-in for the browser that wants to download that. The point of the presentation is that the watermarking that you mention, is not the kind of watermarking that we are familiar with.

Frank Stajano: Well, no, then you would be subject to Mike's argument because someone could remove the watermark and use the picture. Isn't this something that you would object to.

Matt Blaze: They could use the picture, but it won't show up because it won't have a signature.

Michael Roe: The thing about repeating a normal watermark is, the attacker produces something that looks the same to the user but doesn't register

cryptographically. So they can steal it and go and use it. And in the same way, if you break a watermarking scheme in that way then they could produce something that looks like a stamp that says approved for 12 year olds or whatever, but didn't have a cryptographic function. So they can then attach it to a document that didn't have the right properties. So in this application the protection needs to be just as strong as it needed to be for the copy protection.

Roger Needham: I think the thing that's crucial is that you've equipped your browser, or your teacher has equipped your browser, with something which will refuse to display anything that doesn't check out. And then, if you have done that, then it all works. If you change a part of the web page or something, the browser would sense it, for this class of things this is the signature that works out. I mean there may be someone who certifies pornography as being high-class pornography (laughter).

Bruce Christianson: It may have a mark that says, banned by the US Justice Department.

Pictures Can't Lie under Oath

Bruce Christianson, James A. Malcolm, and Brian Robinson

Computer Science Department, University of Hertfordshire
Hatfield, England, Europe

Abstract. Because image manipulation by computer is so easy, it is hard to be sure that an image was indeed created at the date, time and place claimed, and that it has not been altered since. To gain this confidence requires a strong chain of evidence linking the collection of bits representing the image back to the camera that made the image. Cryptographic checksums used to implement parts of this chain must demonstrate that no other image was substituted for the correct one, even though a very long period of time might have been available for a malefactor to exhaustively search for an alternative that matched the checksum. It must be possible to secure the provenance of the image without taking the camera out of circulation, and subsequent tampering with the camera should not cast doubt on the provenance. Signing the image twice, but only publishing one of the public keys, allows the image to be verified whether the camera is available or not, and also undermines any claim that the signature could have been fabricated by trial and error. Watermarking, though largely irrelevant to this discussion, does provide a convenient way of ensuring that the image and the associated data can be handled in a way compatible with existing image processing software.

1 Introduction

Much interest has been shown in retina scanning and other biometric authentication techniques. However one needs to be quite careful about what such techniques prove. If the authentication device is tampered with, or if reference data are intercepted and modified, then a remote system may be misled.

Similar and somewhat more obvious concerns apply when a digital camera is used to produce a photograph to be used in evidence. How does the court know that the image hasn't been doctored? As with conventional photography used in evidence, the concern is with the provenance of the image and the trustworthiness of the photographer. In cases of doubt, with a conventional photograph, the original negative can be produced and examined for evidence of tampering, but with digital images, editing may not leave a trace.

No technique can prevent a criminal asking a friend to take a digitally time-stamped photograph of their twin sibling at the time of a planned crime to use as an alibi should they be caught. In this position paper, we argue that by suitable design of camera we can be confident that the image presented in court was created in the focal plane of the digital camera at the time and date specified and

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 208–211, 2000.

has not been altered since. Whether the image indeed shows what it purports to show is another matter. It is important to state this caveat at the outset, because there is a risk associated with any device that seems so trustworthy as to be completely free from failure. The greater the apparent security of a scheme, the more severe the consequences if someone finds and exploits a loophole.

It is particularly difficult to design a system that can give the court the assurance that a digital photograph has not been tampered with, because legal processes are typically very slow, giving a fraudster plenty of time to modify a photograph and generate whatever signature is required by trial and error. As we shall see, there may be many photographs that are similar to the desired one but which have the same signature as the original.

So we see that we have a chain of evidence problem – showing that a particular picture really had been taken by a particular camera on a given date – together with the problem of finding (and establishing the claims made for) adequate hash functions for image data which will detect significant tampering even when there are many pixels whose value can be set arbitrarily, and lots of time in which to decide how to set those bits. In other words image authentication, in the presence of much redundancy.

2 Outline Design

Briefly, the design we propose is as follows:

The camera has embedded within it the private member of a key pair.

When the digital camera takes a picture, the date, time, and camera serial number (which is the public member of the key pair) are digitally hashed together with the pixels and then signed.

If the camera also included a GPS receiver, we could be assured not only of when the photograph was taken, but also where (if the GPS transmissions have not been jammed, and the receiver is part of the tamper-evident component of the camera).

There should be no way to change pixels, date and time, camera serial number, or camera private key within the camera. The only operation provided by the digital camera should be "take picture".

The camera includes a tamper evident module, and its private key is embedded in that module. On the outside of the module is etched the public key that can be used to verify the authenticity of images produced by this camera (by checking that the hash on the image data matches the encrypted copy generated when the photo was taken).

The tamper evident module should be fail-stop, so that *e.g.* clock failure, flat battery, or other component failure all require the device to be returned to the manufacturer (or other trusted agency) for resetting and recertification.

A paper log of Public Key Certificates is kept, probably by the manufacturer, so that we can know that the camera is not a fake.

3 The Excessive Redundancy of Images

A text message, together with a good N bit hash, requires about 2^N tries to find another message that matches the hash. But text messages are relatively short, and there is not a lot of freedom to generate plausible alternative messages.

The trouble with images is that they are much larger, and that there is colossal redundancy. It would be easy to programme a system to search for images similar to any desired forgery with the correct hash.

There is a very large number of possible pictures. If each image is 1000 by 1000 pixels, in 24 bit colour, there are $2^{24000000}$ possible images. If we make a change to the least significant 1 or 2 bits of each colour value, we still have (to a human observer) the same picture. So there may be as many as $2^{6000000}$ versions of any picture, leaving $2^{18000000}$ distinguishable pictures. In fact things are worse than that, as many of these pictures could be said to show the same event, such as "Grabber Dan entering the Jewellers at about 10:30 on Wednesday morning" (or equally "Grabber Dan on the beach at Bognor at the time in question").

The problem is that we could have an exhaustive search of minor variations of the desired (fake) picture until one matches the hash value (which we know because we have the public key of the camera). We can force any attempted forgery to be repeated for each photo (rather than making it possible to have a set of valid hash values any of which would do for a fake), if the time (and the camera's public key) are used to seed the hashing.

Although this process may not be feasible in practice, how can we convince a court that this has not happened? Relying on mystical properties of hashing algorithms is not sufficiently convincing. There is too much redundancy in a digital image for a straightforward hash function to inspire confidence that Grabber Dan *could not* have generated a collision. We need a stronger indication. (We also need an application specific hash function, but that is a different paper.)

4 A Way Out?

The solution is that the camera in fact carries two public-private key pairs, with only one of the public keys inscribed on the camera back. We can use this key (as described earlier) to be assured that an image is genuine with high confidence (though a strong hash function will still help). The second hash includes the second (secret) public key, so it is a different value from the first, as well as being signed with a different private key.

The other "public" key (which is in fact secret) is kept securely by the manufacturer of the camera (or by some other certifying authority) in a notarised and authenticated log, but can be revealed by court order. Note that if the second public key is revealed, the damage done is not fatal.

This is a form of key escrow which strengthens rather than weakening the security of the system. It guards against two dangers in a system with the whole key on the camera back:

- The person who has possession of the camera knows one public key, so could *perhaps* forge a picture with a checksum that matches the signature of the real image. A defendant should not be able to accuse the police of forging the evidence, and the police shouldn't be able to do so! We assume, of course, that there is no way of extracting either of the private keys.
- Secondly, if the camera is destroyed (not unlikely if a criminal knows that he has been photographed doing the deed), there needs to be a way of verifying the authenticity of a previously secured image. After all either the accused or a bent copper could destroy the camera in order either to cast doubt on the authenticity of the evidence against him or to hide the fact that the evidence has been fabricated.

In both these cases (on either knowing that it is destroyed or on surrender of camera to the court), the court can make an order asking the manufacturer to release the other public key for the camera.

Once the second public key has been released, the manufacturer needs to recertify the module with new keys (else the expensive camera could not be used to take photos usable as evidence again). This also has to be done when the battery goes flat, so that the manufacturer can authenticate the binding of the camera's tamper evident module to the correct date and time. Only the manufacturer will be able to do this – a forged camera will not have a PKC in the manufacturer's paper log.

What we are doing here is binding the photo taken by the camera to the camera itself. The manufacturer binds the camera to the correct date and time, and incidentally, to the name of legal owner.

5 Related Work

There is an interesting contrast between the approach to image integrity which we have outlined here and the watermarking approach.

Watermarking is often used for identifying images in copyright applications. It attempts to bury within the less significant bits of an image, a signal which will be detectably corrupted if the image is manipulated. It spreads an ID, commonly an owner identification, and maybe a checksum or other data, throughout the image.

Maybe a watermark is just a convenient way of carrying some extra data along with an image. Our proposal shows what data should be contained in the watermark. Watermarking can also be used to spread the second (secret) public key through the image prior to hashing, making it harder to tamper with the signed hashes. But the biggest advantage of the watermarking approach is that the image file produced will be compatible with standard software.

Pictures Can't Lie under Oath
(Transcript of Discussion)

James A. Malcolm

University of Hertfordshire

Why did we want to look at digital cameras, what was the thing that made this interesting? The first thing was the link to our previous work: what we've done is an application of what Bruce talked about two years ago, linking bit patterns to real world entities. In this case the bit pattern is the image which has been produced by the digital camera, and the camera itself is the real world entity. Not so much audit trails, but certainly auditing the fact that this image came from this camera, that's the key thing that we want to establish.

There was also a growing awareness of the problem of image editing, the fact that if people were taking pictures with digital cameras, we really have no way of knowing that what they say they took is what actually was taken by the camera. It can easily be changed. And shortly after we began to think about this, Bruce pointed out an article in one of the trade papers of somebody who proposed to use digital watermarking – the police service – in order to be sure that the photographs used in evidence were authentic. And the watermarking was going to be done on the PC in the police station, once the camera had been brought back. Now, I though, this leaves a rather big hole.

So that leads us on to the problem statement. What we wanted to do was to ensure that the image presented in court was what was present on the focal plane of the camera at the time claimed. That's, I think, exactly the same as the situation with a real camera, *i.e.* a conventional handheld camera. We're not trying to do any better than a conventional camera, but we don't want to be any worse either. There's a paper that I discovered recently on the same topic which actually tries to do a little better than a conventional camera, I'll mention that later.

A couple of general points associated with this. The first is, no system is perfect, and if you present a system which is really secure and has really good authentication and all that stuff, then you have to be very careful that both the jury and the court understands that it's no better than a conventional camera and that there are risks. You could get your twin brother to sit on the beach, get a friend to take a picture of him while you're robbing a bank, and that's hard to prevent. So no system is perfect, but the more perfect it appears to be, the bigger the risk that if something does go wrong then the consequences could be serious for somebody.

The other thing that we wanted to try and do was to make a reasonably general solution rather than some kind of specialist solution which would only work for special fancy cameras that the police service might use. Any camera ought to be able to have these facilities, so we don't want it to be too expensive.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 212–218, 2000.

Frank Stajano: You said that nothing is going to be any better than a conventional camera. So are you also claiming that a conventional camera in some sense solved this thing and satisfies at least you with respect to the questions you had on your slide.

Reply: I think so, because if you take a conventional photograph it doesn't tell you when it was taken, but you have the film, you can see the pictures before, the pictures afterwards, you can check that the emulsion has not been tampered with.

Frank Stajano: Where is the time claim proof with the conventional camera.

James Malcolm: We don't have that with a conventional camera, other than the fact that you might take a picture of a newspaper or a famous clock, Big Ben for instance[1].

Bob Morris: If I go now to the camera store and ask for a good camera, will it be digital?

Michael Roe: About half the cameras in their window will be digital.

Matt Blaze: Two questions. First of all, I have never encountered a digital image taken directly from a CCD that didn't actually need to be edited in some way to make it useful. The camera correction has to be done, various fiddling things have to be done with it. That would destroy any signature created with the image, and that seems to be the fundamental problem of any kind of system for this.

Reply: If you had to do any kind of image editing, I think you would need to demonstrate that, here is the original image in its virgin state, it hasn't been edited, you can see basically what the image shows, and somebody has to testify in court that the image editing I've done has only clarified the image, it hasn't changed anything.

Matt Blaze: I think this is probably analogous to printing a negative. Right, you get a negative out of the camera and you make a print of it, of course you're doing some filtering and cropping and so forth.

Bruce Christianson: Sure, but if there's an argument about it, you can always go back, get the negative, establish the provenance of the negative and make another print, a more logical print. In a conventional camera you can go back and check the film, we're securing the original bit pattern, before editing.

Matt Blaze: The second question, I have a vague recollection of reading in the trade press somewhere recently, that someone is producing a commercial product with some kind of digital signature, including authentication, in the camera, but I didn't ...

Reply: I haven't seen that, so you may be right. But just digital signature isn't enough, because then there's this question, OK what keys are being assigned to whom.

Francesco Bergadano: What if you take a picture of another picture?

Reply: That puts us back in the same position. I'm not trying to say that the picture is what it appears to be, I'm trying to say the picture appeared on the focal plane of the camera. What's on the other side of the lens, it may be

[1] Yes, Big Ben is actually the bell not the clockface. We knew that.

real, it may be actors, it may be a painting. I mean, there are various tricks that people can do to try and verify that it's not a painting, but you can do reverse convolutions to make the painting to look just like it would be if distorted by a camera lens.

Matt Blaze: There is one way in which this will not be equivalent. Film is an extremely high-resolution analogue medium. There are essentially analogue techniques for examining whether or not it's degraded in ways that indicate that it is duplicated and so on, there's a large body of expertise there. Whereas a digital camera produces a finite number of bits of output that can always be duplicated, or can probably relatively straightforwardly be duplicated, by putting the correct image in front of the CCD. A picture of the picture, taken on film, can probably be detected, whereas a picture of a picture taken with a digital medium, if it's been carefully generated and carefully positioned, may not be so detectable. But this is an inherent limitation of ...

Reply: I accept that.

Bob Morris: As you speak, will you ever use the word "reality"? That would raise very difficult questions.

Reply: I don't anticipate it. We're concerned with the image inside the camera.

What I'm proposing is that first of all the conventional camera system produces the image on the detector, the CCD or whatever. And there is a tamper-evident component, which contains a processor and a private key, and which will take the image from the detector and turn it into a suitable form for putting in the data store and sign the image. Also within the same tamper-evident component you've got a clock so you know when the image was taken.

So your image is a collection of bits. So you form a collection of bits which can be processed in a standard way, to a standard image, to display software. So this component is tamper-evident, using physical tamper-evident processes. Once you've got the collection of bits in the internal data store, these can be taken out by whatever mechanism you want to use. Eventually it reaches the court computer, and what we require is some tamper-evident mechanism to ensure that these bits and the bits that eventually appear in the court computer are the same, in other words, we haven't changed the image and therefore it is the same image that was formed by the camera.

Stewart Lee: I'm beginning to get a little troubled. I was under the impression that when you went to a court with evidence, the evidence is what someone has seen, and that a printed image that might be presented was only subsidiary evidence to support what someone, for instance, a police officer, claimed he had seen. This contrasts with what you seem to be saying, which is that the picture is the solid evidence and the police officer is presenting a substantiation of the fact.

Ross Anderson: That is what used to be the case. But if you've got a device that generates evidence automatically then Section 69 of the Police and Criminal Evidence Act rules doesn't apply. It's only where the computer has

been used to help someone produce a record that you have to get that person to go along and testify, and this has been the case for some years now.

Bob Morris: The people speaking from the audience here represent three different rules of law, they're significantly different. Canadian and US are very similar with regard to continuity of evidence, but quite different from British law in this respect.

Bruce Christianson: Consider, for example, where pictures taken by a police photographer of a crime scene are being introduced into court proceedings. The issue here goes to the credibility of the evidence, because if a jury can easily be persuaded that the picture has been doctored and the police officer is lying through their teeth, that's not going to help them get a conviction. Whereas a situation where they say, well we *could not* have interfered with this camera, and the jury can be convinced that even a very corrupt policeman couldn't have tampered with the evidence.

Reply: And it works both ways, similarly an honest cop can't be credibly maligned.

OK, so we've physical tamper-evidence here, tamper-evidence using cryptography here, and the public key is actually etched on the back of the tamper-evident components, so there is no way of separating them. The public key is visibly bound to the camera, and so to the image, in order to prove the image shown on the court computer. The court computer is just a standard computer with essentially standard software, and there are lots of them so there's no particular problem with the integrity of this machine because it's a general purpose thing, and you can take the camera-back to the court and show this is the camera with which I took the image. So the loop is closed.

Ross Anderson: Well, presumably what you do is take another picture and observe that it verified with the main public key, because anybody can re-etch the back plate.

Bruce Christianson: It's important that re-etching of the back plate will be detected, that's one of the assumptions.

David Wheeler: You're excluding the possibility that you could duplicate a camera, because you've got the same public key on the back.

Reply: In order to avoid duplicates, and auditing comes in here again, the public keys and the person to whom the camera has been issued are in a paper log maintained by the manufacturer of the camera. So one of the things we're trusting here is that the manufacturer of the camera is doing a reliable job.

Bruce Christianson: Or that whoever is charged with keeping the paper log, is keeping the log.

It's an important presupposition this, that the camera is an expensive piece of equipment. You can't afford to put the camera in an evidence cupboard for the 18 months before the case comes to trial, but equally, if the camera goes out again and is booked out to the same corrupt policeman who has an interest in destroying the evidence chain, you can't allow that to affect your admissibility.

Stewart Lee: So it's the camera giving the evidence as contrasted to the police.

Michael Roe: The camera is an exhibit, it's a piece of physical, forensic evidence.

Bruce Christianson: And the image bits have now been wrapped in a plastic bag.

Reply: We signed the pixels, the bits together with the date and time from the internal clock and the public key, so that we basically got a signed image. The clock cannot be changed, if the battery runs down you have to get the camera recertified, the camera case has to go back to the certification authority, which I'm assuming is the manufacturer but it might be someone different, and basically the only things you can do are to take a picture. Bruce deleted the "discard picture" operation, but ordinary users want to be able to throw pictures away and try again.

Ross Anderson: You don't need to recalibrate when you change the battery. For example the IBM 47/58, they've got the batteries outside the tamper-evident enclosure. There's two battery slots, to change the battery you put the new one in and then pull the old one out.

Bruce Christianson: The assertion is, that if the camera at any point loses power then you need to take it back to the manufacturer. And if any part of the tamper-evident section fails, then you need it to be re-authenticated.

Reply: In effect the the camera is like a public key certificate for itself. It would be possible perhaps to put the global positioning system inside the camera, but I think that's a bit far fetched. And if we did, you could always jam it, so I think we have to stick with the time, that's the only other thing we're certifying.

OK, so it sounds reasonably plausible, but there are some problems. The first problem is image redundancy. You have a colossal number of bits and it's very easy to have another picture that looks, to all intents and purposes, like the same thing, but which is actually a different set of bits. So you could, perhaps, manufacture an image which has got the same checksum, if you have a very long period of time to do it in. Now it's not going to be easy to produce a fake image that matches the checksum, but the lawyers from the other side of the case could claim that that's what you've done. And if you try and use a more complicated hash function in order to emphasise in the hash function more of the significant bits and de-emphasising in the calculation of the hash function the noise in the background, that's just going to make things more complicated.

So, the important thing, we need to inspire confidence that neither the accused nor the police officer could have created a fake image. We want to make it hard to make a fake image, so we want to make the fraud hard, but we also want to make sure that, if somebody were able to create a fake image, albeit unlikely, then we could demonstrate clearly that that had happened.

Bruce Christianson: And that it was fraud, not an unfortunate accident.

Markus Kuhn: Does the quality of the hash function really affect the problem? Germany now has a law that says SHA1 is allowed to be trusted by the legal process for the next six years.

Reply: Well, if I were going to execute someone then I'd want a little more confidence.

Bruce Christianson: But if the law says that an artist's impression drawn by the policeman is sufficient, then that's what the law says and you don't even need a camera. That's a legal question, not technology.

Reply: The solution that we came up with is to use two key pairs, essentially a form of escrow, I'm not sure that escrow is really the right word[2] but we seem to be using it for anywhere you've got a second key. We put both private keys within the camera, and the image is separately signed with both of these. One of the public keys is etched on the back of the camera and is used for verification in normal circumstances. The other public key is kept confidentially by the manufacturer or some certifying authority, and will be used to certify images when there is some dispute. So a court order, or the equivalent in whatever country you come from, can be used to obtain the second *public* key and, without the second public key, the bad guy cannot forge the second signature. He may be able to forge the first signature, but for the second signature there is only going to be a $1/2^N$ chance of randomly picking an image that matches the second signature. The public key to verify the second signature can be obtained only in cases of doubt, the court can request that information.

Bob Morris: You're depending on a trusted third party.

Reply: We are depending on a third party to verify something that we have already checked.

Ross Anderson: There's another mechanism of failstop signatures. A failstop signature – look at Birgit Pfitzman's Ph.D. thesis for details – is a signature with the property that there's a very large number of possible signatures which will be verified with the public key, but only one of them will be manufactured for a given message, given any particular instance of a private key. And so if someone manages to break a discrete log or whatever, and forge a signature, then with enormously high probability they will produce a signature different from the one that the genuine signing key would produce, and you can then repudiate the false signature simply by signing the alleged message with the genuine private key and exhibiting the fact that the two signatures are different.

Reply: This is another mechanism to do the same thing[3].

Bruce Christianson: The difficulty here is that genuine private key may no longer be available because the corrupt plod may have destroyed the camera.

Ross Anderson: Pulled out the battery? Whoops!

Bruce Christianson: So I agree that that mechanism does what you say, but it's not tamper proof.

Reply: The concluding remarks. I started by identifying the problem with watermarking as part of the motivation of this. I think in a way what watermarking can do for us is to find a way of distributing your correctly determined

[2] It is actually the "public" half of the second key which is escrowed and kept secret.

[3] Although it couldn't work in our case, because there must be physically no way to get a bit pattern signed with a genuine private key, except by actually taking the photograph or destroying the camera.

cryptographic information in a way that's compatible to standard image processing software. And what I've done here is to work out what bits need to be hidden in the watermark. The fact that they're hidden is not important, whether they're tightly bound, whether they can be removed or not, is not important. With this application, if you remove the bits, then all you do is remove the ability to verify the image, and what we're trying to do is verify the image.

Francesco Bergadano: The same as Hitachi's.

Bruce Christianson: Exactly. It's very reminiscent of the talk Sasaki gave yesterday. We're taking the two public keys, separately turning them into watermarks of the image and then signing the two watermarks.

Larry Paulson: Why do you need a tamper proof seal between the camera and the court when the bits are digitally signed? If anyone tampers with it ...

Michael Roe: The first tamper-proof seal is provided by physical means, the second one is provided by cryptographic means.

Reply: The first tamper-evident seal may not last, the digital signature *is* the second tamper proof seal.

Bruce Christianson: Well there is still a physical component to the second seal, because there's a register somewhere in the trusted third party's office. But you don't have the same difficulty securing its provenance as you do with a camera that's forged.

Frank Stajano: Because you need a tamper-resistant device anyway you can probably solve the same problem just using symmetric key.

Bruce Christianson: There is then more of the camera that you have to rescue from subsequent interference.

Frank Stajano: Yes, you have to fit some more in the camera.

Reply: One final thing, which may possibly come into the category of an outrageous remark. When we're trying to use this idea of key escrow together with confidentiality, I think the escrow is generally recognised to be weakening the system. I think perhaps what we have here is, when we're using escrow in a system trying to do integrity or authentication, the system is actually given greater security.

Michael Roe: That is a really good, outrageous point to end on, yes, thank you very much.

Bob versus Bob – Good Guy/Bad Guy
(Transcript of Discussion)

Bob Morris

National Computer Security Center

I would like to speak about a number of minor matters that occurred over the past few days that I suggest that you pay more attention to or worry about more or define better.

One is that yesterday morning I several times heard what amount to restatements of the Halting Problem, where people said, hey, we're working on the Halting Problem. It is not necessarily an impossibility to solve this problem but I suggest to you that in many cases it is difficult, time consuming and expensive. For example, saying that a computer program, one which you've probably never seen, enforces a particular policy, may just be a restatement of the Halting Problem. Beware of that kind of thing because there may be a good deal of work down the line in getting things to where you want them to be.

A separate thing I've heard, which I just scratched my head over a bit, was a good deal of discussion of encryption which depends in general on a number of bits somewhere, in some machine, and very strong distinctions were being made between different bits in this encryption scheme, some of them were called bits in the algorithm, some of them were called bits in the key. I'm not sure I know the difference and in many ways I think there is a fuzz between algorithm and key that I have never really been able to get to the bottom of in any satisfactory way. So beware of that, there may not be much difference between the key and the algorithm. Parts are instructions and data that might be used, some of that data might be bits in the key, some might be other things. But that's the second thing that I urge you to be cautious about – where is algorithm and where is data?

The third thing that I noticed was questions of ownership of files. Well, hey, it's partly my fault, but Unix came up with the notion of ownership of files, it crept its way into the Orange Book and into the Canadian criteria and I guess it's into the British too. Unix of course changed underfoot, so now it is not clear what the ownership of the file means in any particular modern version of Unix, or the precautions you should take in dealing with it. To imagine that criteria like the Orange Book have covered that material in any satisfactory way is getting you down a dangerous road. I deal, almost on a daily basis, with three different versions of Unix. Are they all the same? No, they are not all the same. In important ways? Yes, in important ways, and ownership of files is one of those. So I just would like to advise you to follow down any path that starts there cautiously.

In listening to the discussion of the security of thermometers yesterday, that's fine, I've got my own digital thermometers. All the time I was hearing thermometers I was changing it to thermonuclear warheads and getting the same answers

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 219–223, 2000.

except for one. I urge you not to pre-determine the cost and values involved in what you're doing, because you may be dealing with the security of a thermometer, but someone may, very likely will, translate those considerations to the security of the warhead of a thermonuclear weapon. So don't predecide the cost. The cost may show up in very different ways and often do, in different applications in any of the configurations that we do.

Another caution: audit is a very dangerous area, and I have heard that from beginning to end of what I have listened to here, and I have been worried about almost everything I heard. One person yesterday came up with an example that made me happy. I wish they'd raise that again. When you, for example, made a list of user names of people who have tried unsuccessfully to log in to a machine, something terrible happened as a result. What you got is a hell of a lot of passwords. We had tried the same experiment a few years before you did, and I think that we ended up with 85% within a week, of people that had simply gotten out of phase with user name and password. It's a problem that is generally true with audit trails, and you must be exceptionally careful of what you do with auditors. Where they are, who can deal with them, who can read them, who can chase them. I have found very unsatisfactory situations turn up in practice, in actual practice, with the way audit trails are handled, particularly with the standard Unix that came out of Berkeley, with modification of audit trails. If you were doing something on a system that you shouldn't have done, about the simplest thing to do then is basically erase or destroy the audit trail, which is generally very easy to do. It still is actually. So I caution you in dealing with audit trails to be very cautious as you proceed in things like who can read this, who is this for, who is this useful for. This morning, for example, the word "use" appeared right here with respect to audit trails, and they proved that they were useful to some users. Well, yes they are, including me, that's the other me, the bad guy.

I have been paying attention or concerned about four areas of computer security. One that I have largely delegated to others is worrying about the banking system, and in particular the banking system with ATMs which have begun to take over the world. Nowadays if I want some money, hey, perfectly good machine outside, here I just stick the card in the lip and money comes out. Why, I'm not sure, but it does and it's a growing thing. But there are other people here who worry about that to the extent that I don't particularly have to.

The second one that I find that I don't have to concern myself about is some of the privacy questions that come up in computer security, particularly with respect to medical records. It's a nasty and growing area, but I let others worry about that, they're going to bring this up.

Two that I do concern myself with, I've prepared myself to speak about the original topic of this conference, entertainment. One of the jobs that I have not retired from. I am a consultant to a state gambling regulatory agency in the United States and is that interesting, boy, you bet it's interesting. Do they have the problems well under control, no they don't, and the whole area of dealing

with that is a fascinating one that I've continued with. That's the one part, the good guy.

I also thought about preparing a talk about as far as I could get to the opposite of the entertainment industry. Entertainment has an awful lot of bits in it: you have pictures – they've got megabits all over the place – and a lot of sound and whatever. The value of any one particular bit is not great. The value of a lot of them may be great, but any particular one, no. So I looked at an area as far away as I could from the entertainment industry in many of its aspects, and that's another area that I still consult on which is the safety of nuclear weapons. There, in many cases, one bit is all you care about, it really just says yes or no, or now or not yet, or something like that, but it's not really very much in terms of entertainment. But there, as in other areas: I want to recommend in dealing with – particularly and getting into a new area – or going down a new path in some new area, that a very important thing to do right at the outset is to work out who are the good guys and who are the bad guys. This is a very important thing to consider in many or probably all of these areas.

In the case, for example, of good guys and bad guys, you may not be able to work out just what it is. For example, the United States had, say twenty years ago, a large number of nuclear weapons outside the US. A question when we brought them back to the United States: Here are thermonuclear warheads, which we've shipped back to the US. What do you do with them? Where do you put them? How do you protect them? The whole question of who are the good guys and who are the bad guys in times like this is a question that is first difficult to answer and second, absolutely essential to answer. You must work out where they are, or who they are.

Virgil Gligor: I'm more worried about how all the old adversaries are storing those things and managing them than how the West is. How do you propose to solve that problem?

Reply: I spent a great deal of time on that, about two or three years ago. We have been able to solve about 90% of that problem, perhaps a bit more. You're worried about the 10% aren't you?

Virgil Gligor: Well that's maybe 10% too much. It's not negligible, 10%. But what I am suggesting: Gus Simmons if you recall had a protocol for monitoring treaties about 1983/1984 and I was wondering if any technology like that might be useable in this arena.

Reply: The answer is yes. If you carry one of these weapons, in a truck say, down some road, will I know that act? Yes, I will. Almost any interesting place you might go, we will see you do it. Plutonium sings a song. It is a very recognisable song – I'm talking about gamma rays – and you can hear that at the side of the road, no matter how wide the road is, if someone is carrying one of these weapons.

Virgil Gligor: Even if the Earth is on the other side? From a satellite?

Reply: Yes. We have not solved the whole problem but we have worked on it rather hard and we have solved part of the problem – a large part of it, but not enough to make either me or you comfortable. But the question, who are the

good guys and who are the bad guys, remains a very important question that is not very easy to answer. For example, when we had, and still have, nuclear weapons in some foreign countries in Europe and Asia: what viewpoint can we take towards those countries and the government of those countries? That is just a question of identifying overseas who are the good and bad guys. But even after we get those weapons back to the United States, working out who are the good guys and who are the bad guys is not an easy question to answer. But it is absolutely essential to at least provisionally answer that question in all cases.

Now a more interesting area, and much closer to the original subject matter of this conference, is that I have been consulting for some years to a state gambling regulatory authority in the United States. What they do is, they regulate the operations of casinos, and it used to be a very simple kind of thing. All casinos, well they make money for sure, and when they make money they owe a tax to the government, usually the State Government in the United States, and that is some percentage of their take. The first thing you do when you make money is you skim some off the top, then present the rest of the winnings to the State, this is how much we won, and they take their percentage of that, after the skim. And so one of the questions that comes up, what do you do? Do you prevent the skimming? Not a good idea! I would not recommend that. What I would suggest is go to the casino and make your own estimate, which you can do. Count the people and things of that sort, and estimate how much they really made. Then they will report to how much they made, which will be a smaller number. You come to some *a priori* conclusion as to how much skimming you would like to tolerate – I have suggested, for example with the New Jersey Gaming Authority, an amount of skimming that they often view as appropriate – and precede it with taxation, understanding in advance how much they're likely to get. The general view was that 10% is fine, 20% ain't so good. So what do they end up with, well 20% of course. But there again the identification of good guys and bad guys and of what are the competing interests and how do they interact with each other is an interesting choice of question in the gaming area.

I said I didn't deal much with banking clients. I mostly use ATMs just to get money out, I don't think about their security. But here we have a bank that owns the ATM; my own bank, somewhere in the wilds of New Hampshire in the States; and me. We all have interests that to some degree compete and it is not an easy question to satisfy the interests all three parties. For example, at one point I was in Bergen Railway Station, with a train about to leave in half an hour, I put my ATM card in an ATM, and the machine ate the card after I had typed in my PIN. I called up the bank (and my Norwegian is really not that good) and pointed out to them that next to the ATM was a fire axe, and if they wanted their machine they had better show up before my train left otherwise I would get my card back from the ATM machine by the use of the fire axe. They showed up. Well, there's a question of competing interest.

The question of competing interests in casinos has changed a good deal because of two things. One is that we have, what do you call them, slot machines. In the United States the universal term for a slot machine is a one armed bandit.

They're always known that way. They have become computer controlled so that there is some computer inside them that somebody made, and you run some program that somebody wrote to determine how the one armed bandit will act on any particular occasion. If somebody wrote a program that would run on a computer somebody built, and you're suspicious of it – just so. Who knows what Turing would have thought of all of this.

Stewart Lee: It happened it Quebec. The man who made the microchip controller put a trap door in it. He showed up six months later and won $600,000. Then he did the same thing the next week. People became a little concerned. He was investigated thoroughly.

Reply: Is he in jail?

Stewart Lee: Yes.

Reply: Well again, I ask the same question there that I ask about: well, it is a fact that all modern thermonuclear weapons have a chip in, the chip runs a program, the program determines whether, right now, the nuclear weapon will go bang, it determines whether it will detonate. The question I asked when I walked into this was, what chip, who made it, where has it been, who has had control over it, what program runs in it, who wrote it and why did he write what he wrote. Now this may seem paranoiac, you bet, do I exhibit paranoia tendencies in my work, yep, I sure do, and many of these are applicable quite a long way down the line in the sense of what chip is something executing on, who supplied the chip and why, and whoever wrote the program, what were the purposes of whoever wrote the program, and Stew has presented an example, somebody who had some reasons for writing a particular program.

I hope, all I have intended to do is to provide some warnings and some guidance to proceeding down these areas, there are some areas which are rather complex to deal with and you need to take some care. But I would say that first of all, the first thing I look at in any new area is identification of good guys and bad guys. It may seem trivial, it is not trivial. For example, in the case of the ATMs for all practical purposes the bad guys are the banks. Not the crooks, but the banks. Probably the worst of the bad guys are the banks. I mean if you'd asked Mark, you would have gotten some goddamn surprises if you had asked a few more questions. So it doesn't matter, banking, ATMs, nuclear weapons, you absolutely have to know who the bad guys are and who the good guys are, and there may be some surprises and it may be quite unconventional. You will expect, for example, in banking, that when cheating is done it's customers cheating the banks, well, hey, it ain't that, by and large it is banks cheating somebody else, in some cases their customers, more often the government.

Matt Blaze: I would just like to recommend highly, if you're not familiar with it, Steve Bellovin's Analysis of Permissive Action Links, reverse engineering permissive action links from open and unclassified sources. It's on his Web page, `http://research.att.com/~smb`.

Transcript of Concluding Discussion

Michael Roe: I jokingly proposed for a Panel Session, the assertion that there are no new protocols being invented. I think that the last three days have proved me wrong, there are some genuinely different things being done.

For example, in the watermarking discussion we were getting into arguments about whether there are attacks against those mechanisms, but there clearly is a new class of thing being done there that hadn't been thought about in the mid-1980s. It turns that this has a much more fundamental implication, because displaying a bitmap on the screen just didn't exist as an option in those days. We did build protocols that supported or made use of it, but now that we can display bit-mapped images easily, questions arise such as, does it make sense to put a picture of Bob Morris in his certificate rather than the string Bob Morris, to bind a key to an individual. Suddenly these become viable options and these may lead on to a series of novelties over previous attempts.

Similarly, the theme of images reoccurs in James Malcolm's presentation, where the protection of images using existing techniques is an issue and once again he's departed from what we've done previously. And this overlaps with the other theme, which is consumer devices — ubiquitous objects which have microprocessors in. These devices also occurred in Frank Stajano's talk, how do you deal with security, how do they authenticate each other. Authenticating a camera, which is a portable handheld device, is very different from authenticating a mainframe in a bank vault, which is the kind of assumption underlying X.509.

Ross Anderson: There's another set of issues altogether which have to do with computer interaction. Suppose that I wish to sign the photo on my web page saying this is me, I can do so by putting up a file, a giF or whatever, with a PGP signature attached, and someone who is a profound geek can then perform the necessary command line operations to verify signature and assurance that this picture is me. The average windows user can't. What is going to be required is some visual convention which might be, for example, surrounding the photograph with a special crinkly red boundary, such that when you click on the crinkly red boundary, up pops up a window saying, this is the signature and this is the certificate chain and all this good stuff. We see the beginnings of this in browsers, but different browsers — and indeed different versions of the same browser — deal with certificate chains in different ways and have different and interesting bugs. Up until now people have been focussing on the different and interesting bugs, and not on the need to converge ultimately to some reasonably stable set of visual metaphors that people understand and can use. If you bought a car in the 1930s, the brake, accelerator and clutch would be in some essentially random order, by about the 1950s, if you bought a new car there was a convention that whether the steering wheel was at the left or the right the accelerator was on

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 224–227, 2000.

the right, the brake was in the middle and the clutch was on the left. Something like that is going to have to come about and it's going to be a lot more complex than just simply deciding the order.

Matt Blaze: A related issue is distinguishing between an image that is correctly signed, with a valid certificate chain associated with it, and an image whose semantics matches the semantics that you're willing to trust that certificate chain to do.

Bob Morris: Are you talking about reality again?

Matt Blaze: Exactly. That might not be a problem that's solvable with cryptography or for that matter even algorithms.

Virgil Gligor: Well you need a trust policy.

Matt Blaze: Not only a trust policy but you also need to take the human factors component into this, everything like, this is Ross Anderson's picture, and this key is trusted to display Ross Anderson's picture. That has to be somehow associated to the user with the image that's displayed on the CRT. It's a very difficult problem.

Frank Stajano: It looks to me like a self-certification type requirement, because basically a certificate is nothing more than someone asserting something and then putting a signature on this thing. And so if the person who said, this is Ross Anderson's picture, is Ross Anderson, and he puts Ross Anderson's signature under it, then it all boils down to whether you trust Ross Anderson to say the truth about this thing or not. He may put the picture of someone who is a criminal or something like that and then it's basically: if you don't even know him, then why would you trust his signature to sign any statements anyway.

Matt Blaze: I'd like to trust Ross to tell me what his picture is, but that doesn't mean I want to trust Ross to tell me about everything.

Ross Anderson: So you need some means of expressing application syntax, and for better or worse, the one that we're stuck with in 1999 is XML. No doubt there will be something quite different in 2003 because somebody will see a commercial advantage in changing the standards. But whatever the standards are, there is something more to it, it's not just saying that here is the Bank of International Settlements approved XML style-sheet for a cheque. There's some way that you have to communicate this to the user, and have a logo for the Bank of International Settlements that everybody has programmed into their brains, along with some world-wide logo for: this is a cheque. How do you link the symbols together, how do you keep the semiology in step with the systems engineering certificate chains. This is perhaps going to be the hardest work.

Michael Roe: Well we need to say in our report then that at the moment the picture of a key that you get with Netscape is almost useless because it tells you you have a secure session to (pause) someone, and most people won't bother to click on it. As a slight incremental improvement, you can imagine that when I go to the Ross Anderson website and it's signed with his key I get a picture of a red-faced Scotsman superimposed over the key, not only the key, and that's slightly more useable, meaning that ...

Bob Morris: It sounds to me like the search for the historical Ross.

Michael Roe: But at least that gets me out of naming, because what that means is, that at the Certification Authority somebody who looks like this picture walked into their office and handed over a public key, and the corresponding private part signed these bits. And that, as I said, has a certain amount of useful semantics to it.

Bruce Christianson: I think this is an instance of a more general trend. We used to take the narrow view that a security protocol was a conceptual model and a set of message interchanges, and I think that was a purist view rather like the state OSI was in during the seventies. And now we are gradually coming to all the practical realizations of the fact that that view won't fly, that when you specify a security protocol or design a security protocol, you're actually getting into the internal design of the end systems at a very, very intimate level. And that when you specify a security protocol, you have to specify all sorts of state representations and transitions that are not states of a conceptual model but are actually states of the system that you're dealing with, perhaps under some degree of abstraction, they're not neutral with regard to the designer or the implementor of the end system. And I think we've known this for a while but we've been really quite reluctant to come out and face it.

Michael Roe: For example with public key, a great deal of effort in non-repudiation and authentication is, not the bits you sent down the wire, but the surrounding implementations: structures of how keys were protected, what is needed in order to cause the signatures to be computed, what procedures you use to send it by and all that surrounding stuff, and it's not in the message exchange but it's a crucial part of the security semantics of the protocol.

Virgil Gligor: One reason is that in this environment that you are assuming the protocol worked, you also assume that some threats are taken care of by somebody else, so when you try to show or demonstrate what your protocol is good for, you basically have to say what the protocol does and what it assumes somebody else does. So consequently for all these protocols there is a context.

David Wheeler: There's a minor point I'd like to make. We've all been brainwashed into using state, whereas some of our minds tend to move in terms of pathways. They are of course equivalent in some sense, but I think in a new city, some people memorise the map and some people memorise the way they go from one place to another. These are just different ways of looking at the same thing, but I think they do actually give you slightly different viewpoints of some of these discussions. We're always basing them on state, and they are equivalent, but they may not give you the same environmental picture.

Virgil Gligor: One example of that, of something that's not a fundamental distinction just a viewpoint, is this notion of traces that people have. And they express properties as sets of traces. Some properties cannot be expressed as sets of traces, but basically one can prove exactly what you're saying, that the view of a set of traces as being a property is equivalent to a state transition model, in other words, if I can represent something as a state transition model then I can actually represent it as a set of traces.

John McLean, Jim Greis and Stewart Lee took that point of view, that there are some properties that cannot be represented as sets of traces, and they could not have found this result, that information flow is not a safety or a liveness property, without taking the point of view that properties have sets of traces. So it was a very useful point of view in that instance.

Bob Morris: I'm still bothered by that. I think people are telling me that there is little or no distinction between Ross Anderson and a collection of bits.

Michael Roe: Oh, no, we know there is this very indirect chain between the physical Ross Anderson and the bits and we were just wondering about where the indirection goes. We're indirecting that chain through another pattern of bits which is a picture, which is different from our original chain of bits ...

Bob Morris: So you're not saying that Ross is a collection of bits, you're saying that between Ross and a collection of bits there is a chain of evidence.

Virgil Gligor: And whether that chain is real depends on some other things.

Bruce Christianson: Yes. We're trying to design the bit format of Ross's warehouse receipts.

Bob Morris: Fine, I'm happy. So long as I don't have to start thinking about reality.

Michael Roe: Putting reality inside your CPU is bad news because it doesn't fit.

Bruce Christianson: The other way round is better, but still too hard.

Who Knows?

Allan Ahlberg

from "Please Mrs Butler" *

I know
Something you don't know.

No, you don't,
I know it.

You don't know it.
How could you know it!
Nobody knows it,
Only me.

I just know it.

Prove it, then.
Tell me what I know.

Tell yourself.
Why should I tell you?
You're the one
Who knows it.

Yes, but you *don't* know it.

You prove it.

I can't prove it.
How can I prove it?
If I tell you what I know
You'll say you know it already.

I do know it already.

Well, *you* prove it.

No, I can't prove it.
If I tell you what I know
You know,
You'll change it to something
else.

No, I won't.
If you tell me
What you know I know,
I'll know if you know it.

Yes, but I *won't* know!

That's all right.
Then I'll know
Something you don't know.

* pp 20–21 from "Please Mrs Butler" by Allan Ahlberg, Kestrel, London, 1983. Copyright ©Allan Ahlberg, 1983. Reproduced by permission of Penguin Books Ltd. "Please Mrs Butler" is also available in Puffin, ISBN 0-14-031494-6.

M. Roe et al. (Eds.): Security Protocols, LNCS 1796, pp. 228–228, 2000.

Author Index