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

LNCS 5087

Security Protocols

14th International Workshop
Cambridge, UK, March 2006
Revised Selected Papers

Springer

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

Security Protocols

14th International Workshop
Cambridge, UK, March 27-29, 2006
Revised Selected Papers

Springer

Volume Editors

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

Bruno Crispo
Vrije Universiteit, Faculty of Science
Department of Computer Systems
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
E-mail: crispo@cs.vu.nl

James A. Malcolm
University of Hertfordshire, Computer Science Department
Hatfield, AL10 9AB, UK
E-mail: j.a.malcolm@herts.ac.uk

Michael Roe
Microsoft Research Ltd., Roger Needham Building
7 JJ Thomson Avenue, Cambridge, CB3 0FB, UK
E-mail: mroe@microsoft.com

Library of Congress Control Number: 2009935708

CR Subject Classification (1998): E.3, C.2, K.6.5, D.4.6, K.4, F.2

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN	0302-9743
ISBN-10	3-642-04903-6 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-04903-3 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12771844 06/3180 5 4 3 2 1 0

Preface

Welcome back to the International Security Protocols Workshop. Our theme for this, the 14th workshop in the series, is "Putting the Human Back in the Protocol".

We've got into the habit of saying "Of course, Alice and Bob aren't really people. Alice and Bob are actually programs running in some computers." But we build computer systems in order to enable people to interact in accordance with certain social protocols. So if we're serious about system services being end-to-end then, at some level of abstraction, the end points Alice and Bob are human after all. This has certain consequences. We explore some of them in these proceedings, in the hope that this will encourage you to pursue them further. Is Alice talking to the correct stranger?

Our thanks to Sidney Sussex College, Cambridge for the use of their facilities, and to the University of Hertfordshire for lending us several of their staff. Particular thanks once again to Lori Klimaszewska of the University of Cambridge Computing Service for transcribing the audio tapes, and to Virgil Gligor for acting as our advisor.

August 2009

Bruce Christianson
Bruno Crispo
James Malcolm
Michael Roe

Previous Proceedings in This Series

The proceedings of previous International Workshops on Security Protocols have also been published by Springer as Lecture Notes in Computer Science, and are occasionally referred to in the text:

13th Workshop (2005), LNCS 4631, ISBN 3-540-77155-7
12th Workshop (2004), LNCS 3957, ISBN 3-540-40925-4
11th Workshop (2003), LNCS 3364, ISBN 3-540-28389-7
10th Workshop (2002), LNCS 2845, ISBN 3-540-20830-5
9th Workshop (2001), LNCS 2467, ISBN 3-540-44263-4
8th Workshop (2000), LNCS 2133, ISBN 3-540-42566-7
7th Workshop (1999), LNCS 1796, ISBN 3-540-67381-4
6th Workshop (1998), LNCS 1550, ISBN 3-540-65663-4
5th Workshop (1997), LNCS 1361, ISBN 3-540-64040-1
4th Workshop (1996), LNCS 1189, ISBN 3-540-63494-5

Table of Contents

Putting the Human Back in the Protocol
(Transcript of Discussion)

Bruce Christianson

University of Hertfordshire

Hello, everyone, and welcome to the 14th International Security Protocols Workshop. I'm going to start with a quotation from someone who, at least in principle, is in charge of a very different security community than ours:

> Our enemies are innovative and resourceful, and so are we. They never stop thinking about new ways to harm our country and our people, and neither do we.

It occurs to me that if we replace the word "country" by the word "system", and the word "people" by the word "users", then we have a pretty fair description of the current state in our own little security community, the security protocols world.

Our theme this year (may I have the envelope please, Mildred) is "Putting the Human back in the Protocol". We're very used to saying, almost as an afterthought, "Of course, Alice and Bob aren't really people. Really Alice and Bob are programs running in some computer environment." But computer systems exist — the reason we build them is — in order to enable people to interact in accordance with certain social protocols. This means if we're serious about system services being end-to-end (and if we're not, then we can spare the rest of the world an awful lot of what we do) then, at least at some level of abstraction, the end points, Alice and Bob, are human. This has certain consequences.

We're also very fond of saying that when we say, "believes" — for example when we say that Alice believes a particular key is fresh, Alice believes that what Bob says is true — we don't really *mean* believes, we mean squiggle, where squiggle is some mathematical predicate that satisfies certain axioms[1]. But, of course, if the end-points are human, then squiggle does map on to certain beliefs that are actually held by humans. The problem is that currently they're not beliefs about anything useful, indeed they're not the kind of belief that a rational person would willingly hold consciously for any great length of time. The dependent problem from that is that when we invite users to participate in security protocols, they are inevitably the weak link. Usually this is attributed to the fact that humans are untrustworthy, unreliable, and unable to do cryptography in their heads. We've all read Kevin Mitnick's book[2].

[1] Such as KD45.

[2] The Art of Deception, Wiley, 2003.

The trouble is, in ad hoc environments, and particularly in the context of pervasive computing, very frequently the human element of the system is the only one with any real understanding of what the security requirement is. I'm not saying the human knows what's actually going on, but they have some idea of what should be happening, and perhaps more importantly, of what shouldn't. The difficulty is that when we try to program systems to interact with humans by popping up a box which says, do you want to accept this certificate or not, we're asking the wrong question. We're asking questions in terms of the abstractions which we currently use to explain and analyse protocols, and humans are well-known to be very bad at this kind of logical thinking.

But suppose that you take a logical puzzle — does conclusion C follow from premises A and B — and rephrase it in a context where there's some kind of transaction, and the question at the end becomes, is Alice treating Bob fairly or is she cheating him? Now it turns out humans are very good at solving problems posed in this form, even humans who have not spent many years doing post-graduate courses in computer science and philosophy. So the question whether humans are really the weak point is something that I think we should re-examine.

A common objection is also that humans can't do cryptography, but with the increase of personal devices, and the emergence of the pervasive, ubiquitous computing environment, humans are typically now surrounded by a cloud of little devices that can do cryptography perfectly well, and with whom they have an extremely intimate relationship. For "personal" think "unshared", or at least potentially not shared with anybody not trusted.

So the question isn't just, how can we put the human back in the system? This isn't an HCI problem, it's not something that interface people can deal with. The question is, how can we put the human right back in the protocol, how can we align the interests of the human with the protocols that serve them, at all levels. Currently if you look at the deployment of middleware, it's a major research question to try and identify whether two end-points are the same end-point at different levels or not. That's definitely the wrong question to be asking. Perhaps we should instead devote more energy to determine whether Alice is talking to the correct stranger, as my student Jun Li rather nicely puts it[3].

This is a workshop and not a conference. The rules are similar to those of a Quaker meeting, it's OK to interrupt the person who's giving their testimony, but please make sure that your motives are pure, or will at least bear peer scrutiny. We have a few more PhD students than usual, including some from far-off exotic places, which is very nice to see. Please participate, and don't worry if you get it wrong.

These workshops usually descend into chaos at some point, so this year we decided we'd try and just get it over with. Accordingly Matt Blaze has very kindly volunteered to be our first speaker. [Laughter]

[3] Towards a "Localization of Trust" Framework for Pervasive Environments, PhD Thesis, University of Hertfordshire, 2008.

Composing Security Metrics
(Transcript of Discussion)

Matt Blaze

University of Pennsylvania

I have to apologise that, having been asked to set the pace, I have done something inadvertently terrible: I have prepared a presentation and a paper that's approximately in keeping with the theme of the workshop; that is entirely an accident, I have never looked at what the theme of the workshop was, so I apologise for any confusion, please assume that I'm speaking on a completely different topic if you're interested in understanding the theme. I'm going to talk about composing security metrics, and that does have something to do with putting the human back in protocols, and thinking about protocols on a human scale. This is joint work with Sandy Clark, Eric Cronin, Gaurav Shah, and Micah Sherr. Sandy and Eric are here, and this is a result of long conversations, and meetings, and trying to shape something out of what looks like a very difficult subject. We've made very little progress, but we have some pretty pictures.

I'm going to go back to a subject that I've been thinking about for a while, which is how we think about security in the physical non-computational world. I'd like to go back for a moment, and think about the history of metrics for security that is computational but not electronic, and to start with mechanical locks. This is a slide that I used in a talk here a couple of years ago when I talked about human scale security[1]. The people who build locks in the mass produced world had to think about quantifying security from fairly early on, and what that means is that they had to think quite specifically about how their products might be attacked. If you're a buyer of one of these products, or you're a system integrator for these products, you have to think about how much security you need, what's worth paying for, and what isn't. Here is a world in which these things are physical objects made out of material, and locks that are twice as big may be twice as expensive, so you really have to optimise this pretty well, you're encouraged to get just the level of security that you want and no more. And in an environment like that, being able to measure things is an essential first step, even if we don't always measure them exactly as accurately as we might want.

There are a number of different threats you might think about, and there are some interesting analogies back to computer science that I won't talk about. One of the most common ways of compromising physical security is by bypassing the copy protection scheme, and getting an illicit copy of the key that opens the door, for example, you loan somebody a key to do work on your house, and they go to a hardware store and make a copy of it, and come back a month later, and remove

[1] Toward a Broader View of Security Protocols, LNCS 3957, pp. 106–120.

all of your expensive computing equipment. The interesting thing is that this is a problem that is as hard to solve in the physical world as it is in the electronic world, and they've had about as much success as we have at solving it.

Another problem is that the implementation of these locks is not perfect, that is, even though these devices are mass produced and one is similar to another, just as copies of software are similar to other copies, they have defects in them. There are both manufacturing defects, design defects, and overall implementation failures, and they allow things like lock-picking, or brute force attacks against locks (brute force meaning the actual application of brute force, not exhaustive search of the key space). Exhaustive search of the key space is not actually a brute force attack in the physical world, but that's actually a design flaw, right, we can measure the key space of a lock pretty accurately, and figure out whether that key space is sufficient to prevent somebody from trying every possible key; if there are only five different keys to a lock, that's probably not enough.

So let me just speak more about this for a second. By locks I'm talking about the kind of classic pin tumbler lock (although lever locks and combination locks have analogies); if you cut a lock open on the inside you'll see that there are things with security parameters in it, that is, a typical lock contains some collection of tumblers, typically five, or six, or seven, and each of these tumblers has a parameter associated with it, there's typically somewhere between three and ten different positions per tumbler, the key essentially sets each tumbler to a correct position, and if you get the correct position for each tumbler the lock will open for you. Now a nice thing about this is you can quantify keys when you think about a lock in this way, and in fact locks were designed in order to make it possible to enumerate the key space so that you can build different instances of a lock for each of the customers. Essentially you have a position on the key that can be cut to a different height depending on its tumbler's parameter. We can completely describe a key by the name of the manufacturer, and the code for producing the heights on the key, one by one.

Now you can measure the key space, ask how secure are these things against exhaustive search, and do the obvious thing, number of different tumblers raised to the number of parameters on each tumbler, it's exponential, that's good news, computer security says, exponential means secure. They actually have to subtract a little bit because of the physical aspects of this, but as computer scientists we just say, that's implementation, don't worry about it, and it still ends up being approximately exponential. If you plug in the numbers that are actually used by real lock manufacturers a typical lock has somewhere between about 156 (minus a little bit) and 10^7 or so distinct keys.

So, as a computer scientist we'd say, well my Pentium 4 computer can zip through that even at the high end in just a few milliseconds, so these are not secure against exhaustive search, but in fact it's an on-line serial operation, testing each key increases your vulnerability to attack, and so even at the low end of this spectrum exhaustive search is probably not a very fruitful way to attack mechanical locks.

How did people who designed these things long before electricity was available in commercial use, manage to come up with a design that quantifies in such a nice computer science style way when they didn't even have computers? Let me just talk a little bit about history and about how we started to quantify security, and think about how we would compare one mechanical lock to another.

There's an interesting story about two people. One was a British lock maker and inventor named Joseph Bramah, who was the inventor of all sorts of interesting things. He invented the flush toilet, the beer pump, and the high quality mechanical lock, and his lock shop still exists in London. Schoolchildren in Britain often learn about him, Americans don't, we have to use the Internet, but there was an American named Alfred Hobbs who thought he could do better than Bramah. Now for a little over 50 years Bramah had posed a challenge, a precursor to some of the challenges that we now see in computing. If you go to the British Science Museum you can see Bramah's shop; they ripped his Piccadilly shop window out and put it in the basement, there's a nice room devoted to this. The shop still exists in north London, still owned by the Bramah family, but not this particular instance of the shop. Bramah's lock has the nice property that it is beautiful enough that rich people would like to buy it just on its appearance alone, and secure enough that it is adequate for rich people to protect their possessions.

This is a design of a Bramah lock that I purchased a couple of years ago: it is exactly the same as the 1794 Bramah lock basic design, keys from some of the early locks will interoperate with current production Bramah locks, so standards have endured for quite a while here. Although other manufacturers don't interoperate with it, there's no inherent reason for that. In this instance of a lock there are seven parameters arranged in a circle around the keyway, the lock has a circular key that fits in and turns; an interesting property of this lock is that it was designed with interchangeable parts, you can take them apart easily, you can take the tumbler set from one and interchange it with that from another lock, you can change its configuration a little bit, and these were all made out of mass produced parts back at the beginning of the Industrial Revolution, often with a machine Bramah's shop had itself invented.

The Bramah key has essentially the same design as a tubular bicycle lock key, it's just the parameters that have changed a little bit. Dickens mentions Bramah keys in the Pickwick Papers, so, you know, these locks have quite a bit of cachet. The interesting thing about the lock trade is that before Bramah came along, and to a lesser extent, after, but exclusively before, the locksmith was a speciality of the blacksmith. The local ironworker who was good with mechanical objects would fashion a lock, custom made for you. You would buy it from a trustworthy locksmith who you would trust not to sell locks with exactly the same keys to all of your neighbours, and that was pretty much it. There were only one, or two, or three, in a given area, who would work, that work was heavily protected by a trade guild, and it was essentially dependent on the fact that these people had a lot to lose by doing sloppy work, they could get kicked out of their guild, people might not do business with them, and it wasn't easy to move from place to place. In a system like this you needed journeyman credentials that would be hard to get.

Now mass production ruins all this, you eventually get hardware stores selling these things, so how do we know that these locks that are being sold are secure? It's not simply a matter of, we have to trust Bramah, we'd like something a little more substantial. Bramah, by the way, had quite a few apprentices, if you go to the British Science Museum you'll see that some of the rules are similar to being a graduate student today: you had to stay out of houses of ill-fame, taverns, and ale houses, except when you're on business, so it's not quite clear what other businesses Bramah may have been involved with, basically he had a standard form for people working for him, he was really scaling-up lock production, so here we have the first kind of problem of scale.

Bramah quite boldly put a lock in his shop window and said, if you can open this lock non-destructively with any tools you choose to make, I will pay you two hundred guineas, which back in 1794 was real money, and that challenge lock was quite famously put out there. If there had been the Web back there it would have been the first thing on his website saying, this is one serious lock, and I will set up a fair challenge, if you can open this lock I will pay, you must show me how you did it, but I will pay two hundred guineas to the first person who can get this thing open without the key and without brute force, serious enquiries only please.

Now he had an interesting problem: who might respond to a challenge like this? Lock nerds, like the hackers of the day, would obviously be interested in this, this would be a good way of making a name for yourself, but two hundred guineas was large enough money that it might also attract the burglars: that is, those who might want to learn how to attack this lock and keep it secret, might still be tempted to claim the prize because it was about as much money as you could expect to make from a good slew of burglaries, and there's no risk of jail, this is legal to do. So this is a good temptation to get not just the honest people, but the dishonest heavily motivated people to be interested in the challenge. That suggests that this might be a meaningful test of whether this is secure or not, you can either burglarise somebody's home, and maybe come out with nothing, maybe end up in jail, or you could go for this controlled, honest challenge. As far as we can tell, Bramah lived up to his end of the bargain, he would allow you to look at his lock if you asked, he'd let you fiddle with it for as long as you liked. There was some evidence he would actually pay up if you succeeded, and nobody got sued, nobody who discovered any weaknesses in the lock was going to be ruined by lawsuits, there was no digital millennium copyright act as we have in the United States for locks as it were back then. This was an honest challenge.

Fifty years later Alfred Hobbs was able to open Bramah's lock at the great exhibition in London in 1851. The Bramah table was out there, Hobbs showed up, said, let me fiddle with it, he spent fifty hours, and he got it open. Now what do we know as a result of that? Well, it took fifty years, and the person who did it took fifty hours to do it, so we did learn that this lock was in fact not perfect, and we learned also that the Bramah company was still, in spite of producing an imperfect product, probably reasonably honest: they paid up. They didn't try to claim any false things about Hobbs, they basically said, well, you got us, and they gave him his two hundred guineas. Hobbs was a hacker in the proudest tradition of the sense, he focused on this problem for fifty hours non-stop with tools of his own making, and he figured out how to do it.

So what do we have at the end of this? Well we learned a little bit, but we didn't really learn very much. We learned a little bit about an upper bound, and a little bit about a lower bound, but we didn't learn about how this thing performs in practise, we didn't learn what it actually takes under operational conditions to get this thing open. Fifty hours by a determined hacker, well it's still an impressive lock, but on the other hand, we don't know that there isn't somebody much cleverer than Hobbs who's going to come along and figure out, oh, if you just rap it on the side it just opens, we don't know whether that's true or not, we only know that nobody has thought of it yet. So after all this, it's not terribly satisfying to us.

Peter Ryan: This lock was supposed to be the same as the commercially available one?

Reply: Yes, it was the same as the commercially available one, and in fact Bramah didn't change his design as a result of this, and it's still sold today, so again, terribly unsatisfying in the end. But it was the first formalised security challenge. So let's go back to computer science for a moment and think about some of the ways in which we've adapted Bramah's challenge to the modern world. Remember back in the 1990s the US and British governments were trying to claim you only need small keys; other people were claiming, no, we really like big keys for our crypto algorithms. And so people would set up these sort of silly cipher challenges in which they'd encrypt something and challenge people to come up with the decryption key, and offer a prize. This tells us something about whether or not people can marshal enough computational resources to exhaustively search a cipher, or whether there's some shortcut attack.

There are often prizes offered by vendors, very similar to Bramah, in which they say, break my security system, go ahead, I dare you, I'll pay a modest prize. Now invariably these companies are not as honest about this as Bramah, in fact, a pretty good rule of thumb if you look at history is that anyone offering one of these challenges is going to renege on it in some way, you better read the fine print, and check out the laws. In many cases they often offer a prize that's smaller than the expected benefit of attacking a real system, particularly one for financial use, and they may even attack the winner. If you remember the Felton versus Recording Industry case, where there was a DRM challenge, when Ed Felton and his grad students found ways of meeting the challenge in only a few days, the very first thing the challenger did was say, he didn't succeed, now come on, this didn't work, and when it was clear it did work the next thing they did was sue him. So these challenges don't tend to attract people who might be successful at them; there isn't as rich, and as an honourable, a history, as there was back in Bramah's time.

Sometimes attackers will discover an attack, without a formal challenge, a system will just be put out there and researchers will find and publish a weakness in it. Again, vendors of products, as opposed to academics who put out protocols, often try to claim that this is unfair and dishonest, and will attack the attacker. So again, outside of the academic community, finding and publishing an attack is considered to be a kind of dishonest, weird thing to do, and, people are not encouraged to do this in polite company. The case of Diebold versus Avi Rubin, who has been looking at voting machine security, and finding lots of flaws in the Diebold products, is instructive in this regard.

Kenny Paterson: Are you specifically excluding things like RSA challenges from this discussion?

Reply: These RSA challenges which are trying to look at the strength of cipher algorithms have some approximate value to us, but it's not great; they don't tell us as much as we might want to know.

Unfortunately none of these challenges tells us anything in a scientific sense, they don't give us tight bounds on the security, they only tell us whether someone has met the challenge or not.

Tyler Moore: What about iDefense[2] and other companies which are offering to pay hackers for vulnerabilities. They pay out, and they're formalising it to a process.

Reply: Yes, they seem to be trying to do better, but I think the community views these things with suspicion, and justifiably so. Perhaps when more of a history gets developed of actually developing honest challenges, and attracting people who are in the position to find the best attacks, we can draw more conclusions about them than we can draw right now. Right now I think it's safe to say that if nobody has met a challenge for attacking a lot of these systems, we can't really say much of anything useful. And this does not appear likely to change any time soon, particularly with vendors threatening to sue the people who meet the challenges.

Sandy Clark: There was a recent report of somebody trying to report just a bug, not a vulnerability, to Microsoft; not only did it cost them $35 or so, but it took them two days to go through Microsoft's systems simply to report a bug. If there isn't a process available to report a bug, to report a vulnerability, then it's going to be problematic to try and prove that you've actually got something, or to get people to accept that you've found something.

Tyler Moore. Well that's what companies like iDefense are trying to do, they're trying to create a mediation channel for hackers, to pay them for what they discover, and allow them to retain their anonymity. These companies are coming under

[2] `www.idefense.com`

criticism from Microsoft, and other large software companies saying they're causing more harm than good.

Reply: Yes in fact, they're trying to bring hackers into the same fold that we're in, you know, publish these things. Hobbs, by the way, argued for the scientific method about security flaws, he has this wonderfully eloquent statement on that[3].

George Danezis: Isn't the case that you can't say much because the market for attack is actually imperfect. The market for breaking into people's houses and getting the valuables out and selling them for drugs is pretty established, while breaking someone's computer, well it's cool, you know, you can deface it, you can use it as a botnet, but actually selling the information on to another corporation...

Reply: Yes, and it's also quite a fluid world, something that was low value today might be high value tomorrow.

Moving forward from Hobbs, let's go into the world of safes. Between about 1900 and 1950s, the golden age of safecracking, the safecrackers on the bad side were called yeggs, an old term for criminals that was in currency both in Britain and in the US, whereas the safecrackers on the white hat side called themselves safe men (they were invariably male), and had the same skills as the yeggs. An excellent book that I recommend to you, is The Napoleon of Crime[4], about Adam Worth, who was the first safecracker to try and mass-produce safecracking: he tried to do for the criminal side what Bramah tried to do for the building attack side. He was the leader of a criminal ring, and basically he tried to figure out how to train people in the skills, and how to divide labour up in a burglary, and was quite successful at doing so. It's a wonderful biography, out of print but readily available.

Safes have this interesting property that we know quite a bit about how to quantify them because they're used for high value systems. Nobody gets a safe casually, you only get a safe if you think you've got something worth protecting, because they're big, heavy, and expensive. There are two aspects of a safe that one has to worry about, one is the lock for the safe, and the other is the container itself. The locks for safes have this wonderful design, it's very complicated, I've written a paper called, Safecracking for the Computer Scientist[5], where I talk about some of the different quantifiable aspects of safe locks, and how you can go about attacking them, it's easier than you think is the bottom line on that. But, an interesting aspect of this is they're subject to both design weaknesses, and implementation weaknesses.

A safe lock typically has three or four tumblers. One of the implementation weaknesses is that the wheels of the safe might not be the same size, and that may allow a linear attack against what would otherwise be an exponential problem.

[3] `www.crypto.com/hobbs.html`

[4] By Ben MacIntyre, 1997, Delta (US) Flamingo (UK).

[5] U. Penn CIS Department Technical Report. December 2004, see `www.crypto.com/papers/safelocks.pdf`

There are ways to systematically exploit such a flaw, but in spite of that we can build safe lock designs that are designed to resist this for a particular amount of time. So there are different parts of this, they all have to be perfect, but largely aren't, and imperfections can be exploited by the attacker. The standard book on this subject[6] was written by Lentz and Kenton, two safe men who advocated a very different approach from Hobbs, they advocated the anti-scientific method, they suggested after you read their book that you destroy it completely to protect the secret information of it. I got my copy from the library. [Laughter]

So what has evolved since the golden age of the safecracker is that if you go to a safe store and you want to buy a safe, you can find out quite a bit about it, and in fact the insurance industry has largely driven this. You get a rating for your safe that tells you a fairly precise amount of time that it's guaranteed for against different kinds of attackers, so you can buy a ten minute safe, a 20 minute safe, a 30 minute safe, and if you're realty extravagant, a one hour safe, and this is essentially the metric that's used, time to attack. They often will measure this in a couple of different dimensions, the tools that are required, that is, are you worried about somebody who has only hand tools, or no tools, or are you worried about somebody who might have power drills, or are you worried about somebody who can bring explosives in. So a 60 minute explosion-proof safe is big and expensive, whereas a ten minute hand-tool-only-proof safe is smaller and cheaper.

Another metric, by the way, is time versus amount of evidence. You might be concerned, not with whether somebody can penetrate the container, but with whether somebody can penetrate it without leaving obvious evidence of whether it has been attacked. Particularly the containers used for classified information often have the property that they act more as seals than as actual prevention of burglary.

Bruce Christianson: Isn't it better to orthogonalise that problem by having the tamper evident box inside the safe?

Reply: Yes, you could do that, or outside the safe, depending on which order you do encryption or authentication. The question of what you use this for is interesting, and another question is, how were we able to measure it. One reason we're able to measure this sort of thing is that safes mostly haven't changed in the last 100 years or so, they're made out of pretty much the same kinds of materials, better alloys are available now, and the tools have not increased all that much, yes, there have been better explosions, but nobody is using nuclear weapons to attack safes. Occasionally there's a disruptive new tool, like a thermic lance, that's invented, but it's a pretty rare event, so the properties of containers can be pretty reliably measured from a fairly mature engineering discipline which does not exist in computer science.

But ultimately, let's assume we could solve that problem, what you get out of it is this very useful metric, it's exactly what you need to know to measure your system. Time is an incredibly useful metric for a physical security system because

[6] The Art of Manipulation. HPC (reprint edition),1955.

it composes quite beautifully, that is, if you have a safe made out of one inch of material, and you know that will take ten minutes to drill through, two inches of material will take 20 minutes to drill through, we can build a safe twice as good by following simple design rules, so we now how to expand this in a linear way. We also, more importantly, know how to build these things into larger security systems, if I have a 30 minute safe, that tells me an incredibly useful thing if I'm a security engineer trying to design a secure system. If I have a 30 minute safe that is secure against people with drills for 30 minutes, what that means is I need an alarm system that keeps people with drills from having unsupervised access to my safe for longer than 29 minutes. If I meet those conditions, and the metric had been accurately measured, I can have confidence in the security of my system. We can do nothing like that in security protocols and network security systems.

So, why is this? One problem is that we don't try to build 30 minute crypto protocols, we don't try to build security protocols that do anything like this, we try to build things that are perfect. If we can figure out how to attack it in 30 minutes, we consider it to be completely insecure, so we don't even *aim* to do things like that. Unfortunately, this leads to a situation in which we build systems with completely unknown properties, because all we can actually say in practice, is that no attack has been found yet that is better than infinite. But I don't know whether or not such an attack exists, all I know is one hasn't been found, and by the way, I don't have a good mechanism for finding it, because I sue people who tell me about things I don't want to hear. We also don't understand attackers and their tools in nearly as good a way. Disruptive new attack technologies are discovered all the time, that's one of the things we're here trying to do.

Another problem is Moore's law, because it says that metrics that are based on computational capacity (unless that computational minimum attack capacity is very, very large) are not very useful to us because they're going to become invalid after a short time. So what we end up with are three classes. Even if we can measure something as precisely as (say) order of 2^{45}, what we really do is put things into three buckets: effectively infinite security, although we don't actually have that, almost no security, which means that an attack has been found, or, unknown security, which is the common case. Effectively infinite security is achieved pretty much only in crypto algorithms, it's never achieved in implementations, and often we're wrong about whether or not a crypto algorithm is as secure as we think. These metrics, by the way, infinite, almost zero, and unknown, compose in ways that are non algebraic: double infinite is still infinite, double zero is still zero, but more importantly, on doubling unknown is still unknown, but then you get weird things about additions, so infinite plus almost zero is almost zero: adding a mechanism to a system may increase its security, or it may decrease its security, we don't know. Adding an unknown security mechanism to an infinitely secure security mechanism, like a crypto mechanism, leaves you with unknown security. So, you know, we don't know how to compose these things, all we know how to do is measure mechanisms to give us these three unsatisfactory categories, and we have no idea what to do with those numbers when we get them, so we've got a lot of work to do.

What I would like to propose, and in this talk I'm only giving problems, not solutions, is that we need metrics that consider things other than whether or not an attacker is going to out-compute us: we need to find things that don't change, that measure the attacker, measure the protocol, and measure the system as a whole, and we need to do this in a way that system designers can actually make use of this. Measuring the implementation quality, is there a buffer overflow in this system, is not likely to be as helpful as we'd like it to be, measuring implementation qualities has been the $P = NP$ of software engineering, it has been an open problem for a while, we don't known how to do it, and we should not build security mechanisms that depend on the software engineering community's ability to teach people how to build and measure the quality of software.

So, how might we proceed? We don't know how to measure protocols, we don't know how to measure implementations, and we don't know how to compose anything, this is good news for the researcher, it means any progress we make is progress. So let me propose a problem where we might make some progress, which is network denial of service. Given Moore's law we don't want to build denial of service prevention mechanisms that measure how resistant they are to an attacker's computational capacity, because that's just going up. Same thing applies to bandwidth...

Tuomas Aura: What about that as a ratio of that to the computational capacity?

Reply: Well, yes, exactly. This is an interesting question, maybe we can measure the bandwidth of the attacker, and the number of hosts the attacker controls, that's also going up thanks to something similar to Moore's law, but what about ratios is precisely the right question to ask. For example, if you look at the percentage of the Internet an attacker may be able to compromise, it appears to be approximately constant. Can we build mechanisms that are secure against attackers who can compromise, at most, one tenth of one percent of the hosts on the Internet? That is likely to be a moving target for the attacker at the same rate that it is a moving target for the defender. Maybe there are other mechanisms like this, but I think this is what we should be looking for.

My purpose here in taking you down this kind of historical diversion about safes is to encourage you to think along those sorts of lines. I think I have managed to generously exceed my time. Thank you.

Putting the Human Back in Voting Protocols

Peter Y.A. Ryan and Thea Peacock

School of Computing Science, University of Newcastle,
Newcastle upon Tyne, NE1 7RU, United Kingdom

Abstract. Cryptographic voting schemes strive to provide high assurance of accuracy and secrecy with minimal trust assumptions, in particular, avoiding the need to trust software, hardware, suppliers, officials etc. Ideally we would like to make a voting process as transparent as possible and so base our assurance purely on the vigilance of the electorate at large, via suitable cryptographic algorithms and protocols. However, it is important to recognize that election systems are above all socio-technical systems: they must be usable by the electorate at large. As a result, it may be necessary to trade-off technical perfection against simplicity and usability. We illustrate this tension via design decisions in the Prêt à Voter scheme.

1 Introduction

The trustworthiness of voting systems and technologies has received a high level of media attention of late with problems occurring in, for example, the recent US presidential elections and UK postal voting trials. Many of these problems stem from the "black box" nature of the systems, and the fact that they must be trusted to function correctly and as intended.

Considerable progress has been made in the last few years in developing cryptographic voting schemes. These typically provide impressive technical properties such as ballot secrecy, universal verifiability (anyone can verify the correctness of the outcome) and unconditional integrity (integrity is guaranteed without requiring any computational assumptions). Whilst many of these are marvels of the cryptographer's craft, they are typically unsuitable for real elections, in particular general elections. The subtle mathematical arguments justifying the trustworthiness of such schemes are beyond the understanding of electorate at large, even stretching the capabilities of trained mathematicians. In addition, they often involve quite complex interactions between the users, *i.e.* the voters and officials, and the system, and so are prone to error and "social engineering" style attacks.

As with all "secure" systems, even a technically superb voting system may be prone to failure due to human fallibility. For many secure systems we are content to trust in security officers to bear much of the burden of maintaining the system security. With voting systems, by contrast, our goal is to avoid the need to place such trust in a small number of officials. Instead, we are seeking to enable the voters to contribute to the trustworthiness of the system. Ideally we would like the trust to reside solely with the electorate: "dependability by the people for the people!" We thus have the rather paradoxical situation of wanting, on the one hand, to make the voting ceremony as simple as possible, allowing the voter to "vote and go" in the jargon, whilst, on the other, arranging for the voters to play an active rôle in maintaining the dependability of the system.

We outline the goals and key features of a number of voting schemes and describe some of their system-based failures modes. We then discuss attempts to design schemes to take account of the rôle of the human users and strike the right balance between technical and social enforcement of the security requirements.

2 Voter Verifiable Elections

A voting system is a highly adversarial system. Potentially, voters are trying to cheat the system, the system is trying to cheat the voters, coercers and vote buyers are trying to influence the voters and voters are trying to fool the coercers. This last form of cheating is one that we want to encourage, or at least enable. The ability to cheat the coercer means that any proof constructed by the voter will not convince the coercer of how she voted. Ideally, we would like to develop a system in which nobody has to trust anyone. More precisely, we would like the trust ultimately to rest on the electorate themselves. Of course the electorate could set up a large collusion to corrupt the system, but there would be little point. Presumably, the outcome would be democratic anyway, as long as the collusion set has the majority!

Significant progress has been made recently in the development of voting systems with remarkable technical properties such as universal verifiability, coercion-resistance, and minimal dependence on system components. Some of these treat the problem as a special case of the problem of distributed, secure computation, and as such, involve some fairly forbidding mathematics, and tend not to scale well.

A rather different approach, exemplified by the voter-verifiable schemes of Chaum [1] and Neff [2], [3] and Prêt à Voter [4], strives toward schemes that, whilst achieving similar goals, are more practical and accessible. These provide the voter with an encrypted receipt which the voter can later use to check that their receipt is entered correctly into the decryption/tabulation phase via a secure Web bulletin board (WBB). However, all of these schemes harbour certain system-based vulnerabilities. Karlof *et al.* have identified some of these vulnerabilities in an analysis of the Chaum and Neff schemes [5]. Ryan *et al.* have carried out a similar analysis on Prêt à Voter [6]. Some of these can be thought of as "social engineering" style attacks, in which the vote capture device induces the voter to follow the protocol steps in an altered sequence. For example, the "cut-and-choose" element of the protocol could potentially be turned into a "choose-and-cut", thus allowing vote corruption to go undetected. Alternatively, the device could feign an abort if the voter makes the "wrong" choice and repeat the protocol until the voter gets it "right".

Of course, alert voters with a good appreciation of the rationale for the mechanisms involved in the voting protocol would presumably detect and report attempts by the device to deviate from the proper running of the protocol. Unfortunately it is not clear how much we could rely on such vigilance on the part of the voters.

We can illustrate the tension between trying to make the voter experience as simple as possible on the one hand, whilst trying on the other to minimize the system-based vulnerabilities, by reference to a design choice in the Prêt à Voter. The key innovation of the Prêt à Voter scheme is to use ballot forms for which the candidate order is randomized. Information allowing the tellers to reconstruct the permutation, and hence extract the vote value, is buried cryptographically on the ballot forms. In effect, the frame of reference in which the vote is encoded is randomized. Consequently, there is no need to directly encrypt the voter's selection and hence no need for the vote capture device to learn the voter's selection.

In the polling station, the voter selects a ballot form at random. In practice, these would be kept in sealed envelopes. In the booth, the voter extracts the form, makes their mark against their choice of candidate, and then detaches and destroys the left hand column that carries the candidate list. This leaves a receipt of the form shown in figure 2, which in this instance encodes a vote for Democritus.

It is essential for the accuracy of the tabulation to ensure that, for each ballot form, the cryptographic values accurately reflect the candidate permutation shown on the form. An example is shown in Figure 1 below. In the original Prêt à Voter [7], checking the well-formedness of the ballot forms is achieved using a "cut-and-choose" mechanism. In essence two permutations along with corresponding crypto values are given per ballot form. The voter makes a random choice with which to cast their vote. The permutation against which the voter makes their mark is destroyed, whilst the unused one is preserved, and can subsequently be used to check that the permutation is correct. An alternative approach, adopted in the later version of Prêt à Voter [4], is to use a

Democritus	
Plato	
Socrates	
Thales	
	$7rJ94K$

Fig. 1. A typical Prêt à Voter ballot form

X
$7rJ94K$

Fig. 2. A Prêt à Voter ballot receipt

single permutation on each (pre-printed) ballot form, and use random audits of these forms to detect any attempts to decouple the candidate permutations and crypto values. In essence, the “cut-and-choose” element is separated out from the vote casting protocol and is performed by independent auditing authorities in advance, rather than by the voters themselves.

The first approach of [7], which is closer in spirit to Chaum’s original scheme, enables “on demand” creation of ballot material and does not depend on assumptions about the probity of the authorities, or procedures in performing the random audits. It is however more vulnerable to the social engineering-style attacks mentioned earlier, and depends on the voters making unpredictable choices during the vote casting protocol and being reasonably diligent in checking their forms.

An example of the kind of social-engineering attack that might occur in such a scheme is for the device to fool the voter about the sequence of interactions. If the device can predict which side the voter will choose to cast their and vote and which will be audited then the purpose of the “cut-and-choose” is undermined.

All of this might suggest that the most robust implementation is to combine the two approaches. In fact, this doesn’t quite work out: whilst we do get the best of both approaches we also get the worst. In particular we have the problem that the pre-auditing approach requires prior commitment to the ballot material which also opens up certain system-based vulnerabilities, such as chain of custody issues and chain-voting [6]. In the chain-voting attack, an outsider obtains an unused ballot form, marks his vote choice and persuades a voter to cast it at the polling station. If the voter returns with a fresh ballot form, the process can be repeated with another voter.

A possible approach is to use a two sided, three column ballot form:

Democritus		--------------
Plato		--------------
Socrates		--------------
Thales		--------------
	$7rJ94K$	--------------

Fig. 3. Prêt à Voter ballot form: side 1

Figure 3 shows one side of such a dual ballot form, whilst figure 4 shows the flip side, “side 2”. These two sides should be thought of as rotated around a vertical axis. Note that each side has an independent randomization of the candidate order along with the corresponding cryptographic values. Thus each side carries an independent Prêt à Voter ballot form.

The voter uses only one side to encode their vote and makes an arbitrary choice between the sides. Suppose that the voter in this case chooses what we are referring to as side 2 and wants to cast a vote for “Thales”. They place an X against Thales on side 2 and then destroy the left hand strip that shows the candidate order for side 2. This results in a ballot receipt of the form:

The voter’s choice is now encoded on “side 2” of the receipt. Notice that destroying the left hand column of the chosen side also destroys the blank column

Democritus		——————-
Thales		——————-
Plato		——————-
Socrates		——————-
	$Yu78gf$	——————-

Fig. 4. Prêt à Voter ballot form: side 2

Democritus	
Plato	
Socrates	
Thales	
	$7rJ94K$

Fig. 5. Prêt à Voter ballot receipt: auditable side

	——————-
X	——————-
	——————-
	——————-
$Yu78gf$	——————-

Fig. 6. Prêt à Voter ballot receipt: vote encoding side

of the flip, audit side leaving the candidate order intact. The audit side does not contain any information about the voter's selection but the candidate order is still visible along with the corresponding crypto value. Since the permutations of the candidate list on the two sides are wholly independent the voter's mark on one side is unrelated to the candidate order shown on the other.

At the time of casting the vote, the information on both sides of the resulting receipt is recorded and, after the close of polls, would be posted to the WBB. Whilst the information on the flip side conveys nothing about the voter choice, it can be used to check the well-formedness of (the unused side of) the ballot form. For this, the auditors require the seed value for the audit side to be revealed and recompute the encryption and candidate order. It can then be checked that these recomputed values agree with those shown on the form. Mechanisms are needed to prevent the seeds for voted sides being revealed, and is a topic for future investigation.

This is very close in spirit to Chaum's original scheme but with the extra feature that we are now introducing the idea of well-formedness checks on the material on the WBB. This was actually possible in Chaum's original scheme but seems not to have been proposed. Chaum's scheme proposed the idea of voters using checking devices provided by independent authorities on the way out of the polling station. The same could also be done here of course as an extra layer of security and a way to pick up problems earlier.

This scheme has the appealing feature that the two sides are essentially equivalent and hence there should be no voter bias between them. Given that we have the "cut-and-choose" protocol with post-auditing, such ballot forms could be printed on demand in the booth. Any attempt by the device to corrupt votes by incorrectly printing the forms would with high probability be detected. The downside is that it is important that the voters understand the process sufficiently. For example, it is essential that the device not be able to influence or predict the voter's choice of side with which to encode their vote. It is also important that they appreciate that they should only mark the chosen side and that the LH strip of the chosen side should be destroyed. Strictly speaking, any mark on the unused side should not matter, nothing will be counted from this side, but there may be psychological implications for the voters.

It may be possible to automate the above protocol or enforce it procedurally but of course this would require transferring trust to the devices or processes that perform the enforcement.

3 Conclusion

In [8], Anderson shows that cryptographic systems typically fail not due to technical failures but as a result of crude system-based failures. This observation is, if anything, even more valid when applied to voting systems. They are required to be usable by the entire electorate. Furthermore they are used only infrequently so we can we assume little in terms of user familiarity and understanding. On the other hand, we would like the trustworthiness of our voting system to rest ultimately on the electorate.

We have illustrated this tension with a concrete example of a design decision that arises in exploring possible implementations of the Prêt à Voter concept. On the one hand we could opt for a design that makes the voter experience very simple and familiar but requires some degree of trust in the authorities that perform the random audits of the ballot forms. On the other hand, we gave an implementation that removes the need for such trust, the voters perform the auditing, but at the cost of making the voting experience slightly more complex: the voters have to make an arbitrary choice between sides of the form. etc. Either choice has its dangers: in the first a large scale collusion of auditors could undermine the integrity of the election along with the problems of chain of custody of pre-printed forms. In the second, the danger is that voters may not be sufficiently aware of the security mechanisms and so may be prey to social engineering style attacks.

Thus, in designing voting systems for "real" use, it is essential that account be taken of the rôle of the human. It is often claimed that the users are the weakest link in a secure system's defences. All too often this is true, but in the context of voting systems, we are seeking to make the users the bedrock on which the assurance rests. A delicate balance must be struck between making the voter's rôle as simple as possible and enabling the voters to contribute to the overall dependability of the system.

References

1. Chaum, D.: Secret-Ballot Receipts: True Voter-Verifiable Elections. IEEE Security and Privacy 2(1), 38–47 (2004)
2. Neff, A.: A verifiable secret shuffle and its application to e-voting. In: Conference on Computer and Communications Security, pp. 116–125. ACM, New York (2001)
3. Neff, A.: Practical high certainty intent verification for encrypted votes (2004), http://www.votehere.net/documentation/vhti
4. Chaum, D., Ryan, P., Schneider, S.: A practical, voter-verifiable election scheme. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 118–139. Springer, Heidelberg (2005)
5. Karlof, C., Sastry, N., Wagner, D.: Cryptographic voting protocols: A systems perspective. In: Proceedings of 14th USENIX Security Symposium, pp. 33–50. USENIX Association (2005)
6. Ryan, P., Peacock, T.: Prêt à voter: a systems perspective. Technical Report CS-TR-929. University of Newcastle upon Tyne (2005)
7. Ryan, P.: A Variant of the Chaum Voting Scheme. Technical Report CS-TR-864. University of Newcastle upon Tyne (2004)
8. Anderson, R.: Why Cryptosystems Fail. In: Conference on Computer and Communications Security. ACM, New York (1993)

Putting the Human Back in Voting Protocols
(Transcript of Discussion)

Peter Y.A. Ryan

School of Computing Science, Newcastle University

I'd like to talk about the role of the human in voting protocols. Basically I want to argue that voting protocols seem to be particularly interesting from the point of view of the theme of this workshop, in the sense that the users of the system actually play a particularly important role in trying to maintain the assurance of the system itself. I'm interested in a particular class of voting protocols, so-called voter verifiable schemes, which aim to allow the voter to play an active role in contributing to the dependability and assurance of the system. In designing these systems, clearly that we want high assurance of accuracy, but on the other hand we have to balance that with maintaining the ballot secrecy, so that nobody can work out which way a particular individual voter voted, and we want to do it in such a way that we place minimal, or ideally, zero trust in components, such as, hardware, software, the voting officials, and so on, and suppliers.

There are some down sides to the approach that we've proposed. This kind of authority that we've postulated has to be trusted; not for accuracy, because the auditing means that we don't have to trust them for accuracy, but we still have to trust them for secrecy, to keep this material that they've created secret. We need to place some trust in the auditors; now arguably, of course, the trust is minimal, particularly if we've got a set of conflicting hostile auditors, we can spread the trust. We also have a chain of custody issues, we have to keep this material secret, because otherwise we get into chain voting coercion type attacks if some third party can get hold of the ballot form material ahead of time.

Bruce Christianson: Can you describe what you mean by a chain?

Reply: The idea actually works with conventional paper ballot forms, but it's particularly virulent here. The idea is that some adversary coercer gets hold of a blank form, the coercer marks it with a choice of candidate they want and as voters are coming into the polling station the coercer presents the voter with one of these forms and says, if you come out with a blank form I'll give you ten bucks, or whatever it is. Once you've started this, as long as the voters go along, you can continue indefinitely into the future. In the UK system, the voter registers, gets a fresh form, and then they supposedly go off to the booth and mark it, and cast it. And in this attack, the voter is being induced to cast the ballot form which was marked by the coercer, and come out with a fresh ballot form that they were given when they registered, and this kind of attack is particularly virulent here because of the web bulletin board. So here the danger would be if a coercer managed to get hold of one of these forms, and so they know the association of candidate list and cryptographic number variable on the ballot form, they can of course later visit the web bulletin board and check that the voter really did use that ballot form and mark the cross in the correct position. Does that make sense?

Michael Roe: So for example if the coercer goes to the polling station first, has some piece of paper about the right shape with them, and by sleight of hand puts it in the ballot box, that means he's still got in his pocket the real ballot form, to start the chain.

Reply: Yes, so that's how you would initialise that kind of attack, or you bribe an official or something. There's always some way you can initialise the attack.

Bruno Crispo: This kind of attack is very peculiar to the UK, because in another place you cannot do that.

Reply: It turns out in France and Greece there's a different system. The problem with the UK system is that the ballot forms are a controlled resource, and so if you come out with a blank form, that suggests to the coercer that you did cast the marked form. Now as I understand it, in France there's a difference; the ballot forms basically lie around, and you actually register at the point that you cast your vote. Is that right George?

George Danezis: Yes.

Reply: Which actually helps, certainly with the conventional pen and paper systems. But actually if you think about it, it doesn't really help you with these cryptographic schemes, because the attack still works since the ballot forms have unique numbers, etc, etc. So there is a countermeasure which works for ordinary pen and paper systems, but it fails for these cryptographic systems. Perhaps we should distribute the creation of the ballot forms in such a way that no single entity knows the secrets, and in such a way that we just reveal, say, the candidate list, at the last moment in the booth so as to avoid these chain of custody, chain voting, attacks. In a sense, we want on-demand creation of the ballot forms, in the booth, so only the voter sees the candidate list at the time that they need it, and the candidate list then should get destroyed immediately they've made their mark on the form.

The difficulty now is: how do we ensure that the device in the booth shows the voter the correct candidate list? Previously we solved that problem by having the pre-commitments and the pre-auditing of the ballot forms, if we're doing an on-demand creation, of course, we can't exploit that anymore.

So that seems to suggest, as far as I can see, that we have to go back to a kind of "cut and choose", or similar tactic, to try and detect any corruption by the device in the booth. There seem to be several possibilities, but I'll just describe one here, which is actually to print double-sided ballot forms in the booth. So in effect these are two independent Prêt à Voter forms that you saw earlier, but printed on flip sides of a sheet. The voter can randomly choose one side to use to cast their vote, so let's suppose in this case the voter's chosen the left-hand side, they've put a cross against Plato. They should leave the other side untouched, and in accordance with the previous system we again have to destroy the chosen side to cast. So we end up with a receipt which has the two sides; the left-hand side is the receipt of the form that you saw earlier, but on the flip side we set things up in such a way that the ballot form remains intact with the candidate list, and the crypto material here, and the point of doing that is that this can then be audited and checked at a later stage, or later stages, in fact. So the device, if it were trying to cheat, would in some sense have to predict which side the voter was going to choose, and perhaps corrupt that side, and then make sure that the other audited side was correctly constructed.

Bruce Christianson: Does this give me a 50% chance if I'm trying to corrupt the system?

Reply: Of buying the vote? I don't think it helps with buying. The point about this "cut and choose" thing depends what kind of attack you're worried about. There's privacy and accuracy, and this is primarily against accuracy, trying to guard against the device being able, in an undetected way, to corrupt the construction of the ballot forms in such a way that the vote would be incorrectly decrypted at the end.

Bruce Christianson: Ah right, I understand, if it's doing it systematically…

Reply: Yes, that's right, in each case it's got a 50-50 chance of getting away undetected, but of course if it tries multiple times, it falls off exponentially.

Matt Blaze: So the receipt you walk out of the booth with, and the cryptographic thing on the other side, that number of the second ballot, that doesn't help them construct a new ballot to vote again with. The Plato ballot is invalid?

Reply: This should get invalidated, yes. There are issues about how you make sure that ballots can't be recast, and how, for example, you make sure that you can only audit the correct side, because of course you don't want to start revealing seed information. So there are issues there.

Matt Blaze: So, you can't use this receipt, in an obvious way, to learn who someone voted for, but you can learn whether, for example, they submitted a valid ballot. Can I pay people for spoilt ballots, with two Xs, if I want to pay people to *not* vote?

Reply: That's an interesting point, and yes, people do worry about that; this is a sort of coercion to abstain. An issue is whether you have some sort of device in the booth to kind of assist the voter in this process, because this is now getting to be a more subtle process than I described previously. So you could, in theory, have a device which allows you to put just one cross in one side and which automatically destroys the left-hand side. That might help address those kinds of problem, but of course it throws up a whole host of other problems: do you trust this thing which assists the voter, and so forth. In a sense the message I'm trying to get across is there are trade-offs that we have to play against here. And in the remote context things shift again, the issues are completely different, and there's a different sort of answer to your problem.

James Heather: But you can coerce somebody into not voting anyway because you can coerce somebody into not turning up to the ballot station. You don't have to force somebody to cast a spoiled paper.

Matt Blaze: But that's harder because I have to watch you all day as opposed to simply paying you.

Bruce Christianson: You can imagine adding a candidate called "none of the above" to the ballot paper. Then you can tell the difference between someone who's exercised their democratic right not to cast a vote, and someone who has spoiled the paper.

Chris Mitchell: It doesn't help to persuade people not to turn up if they're legally obliged to vote, because in order to coerce them into not voting, you do need a mechanism to be able to discover whether they've deliberately abstained.

James Heather: In places where you're legally obliged to vote, are you are still allowed to spoil the paper?

Bruce Christianson: Yes, although in Australia, where every citizen is legally obliged to vote, candidates whose names begin with A are greatly sought by political parties.

Reply: Incidentally, that's a sort spin-off advantage of this kind of scheme because everything's randomised each time.

It seems to me voting systems are peculiar in the sense that we really would like to set things up so that the assurance arises ultimately from the users themselves. Whereas we tend to think of the users as being the weak link, here we're trying to build the whole structure of trust on this bed of sand, the weak links, the voters themselves. That makes things very tricky, particularly with the electorate at large: just how much can we trust their understanding and diligence in executing the protocols? So it really is a very challenging sort of design problem.

Bruce Christianson: If you can get this politician elected we will pay you two hundred guineas.

James Malcolm: We're all very used to systems becoming more and more complicated, and more and more difficult to understand. Are there, following on from Matt's talk[1], any ways you can see in which to measure and evaluate the trade-offs and decide whether it worth putting in this extra complexity?

Reply: Well, that's a good question. I suspect you probably will have to run trials to try and evaluate voter reaction to some of these trade-offs. And there's a whole host of other issues surrounding the extent to which will people understand and trust the systems, and be prepared to use them. And there are issues about metrics as well, as prompted by Matt's comment, which I think are going

[1] Blaze, these proceedings.

to be particularly challenging with voting systems because it's a whole socio-technical system here. Quite how you evaluate the possibility of certain patterns of collusion, for example, in a metric, is going to be really tricky, so there's a whole bunch of interesting challenges here. I don't quite know how you evaluate these things, it's a major issue and I'd welcome ideas.

Mike Bond: On the idea of this trade-off between what benefit you get vis-a-vis the extra complexity of that, my opinion is that building coercion resistance ultimately is going to be entering into an arms race which the voting scheme manufacturers are always going to lose. The reason is the coercers needn't actually base their coercion method on established fact and truth, they can say to the voters, we've got a secret camera in the voting booth which will see where you put your X, even if there isn't one, and every time there's an election they can make a different story, this year it's a camera, next year it's cryptographic, and the year after is a magic box, or a talking dog that hides underneath the table that can hear which way they're voting. If the voters aren't educated enough to understand there's no such thing as a talking dog, then they're always going to lose this race, and the people who are attacking can try a different attack every year, and only reveal it afterwards. The people who are defending have got to try and educate the voters to call the bluff of every coercionist, so I think it's fundamentally a race that can't be won by the defenders. So I would say that adding complexity to improve coercion resistance is probably not very worthwhile. But improving accuracy, definitely.

Reply: Well we're targeting both.

Bruce Christianson: What we've got to do is to demonstrate that we have put countermeasures to these imaginary attacks into a protocol.

James Malcolm: Imaginary countermeasures?

Bruce Christianson: Well, possibly. But the real threat is that the imaginary attack is credible. That needs a real countermeasure[2].

Reply: This idea of giving voters encrypted receipts, and allowing them to check that they've been entered into the process, seems intellectually very appealing, but it may ultimately flounder on precisely that sort of issue, because you can have these con tricks where someone claims that they can read all of them.

Mike Bond: I suppose it's hearts and minds, because people could say, well it uses complicated cryptography, my vote must be secure, even if the coercer says that they can see it.

Bruce Christianson: It's as safe as your money is in an ATM. Is it possible to get schemes of this kind to scale linearly with fraud? What you hope is that buying ten votes is going to be ten times as difficult, or ten times as complicated as buying one vote, whereas with a lot of schemes there's a magic key, or some sort of magic point of failure, and once you've got that, you can commit fraud on an industrial scale.

[2] cf. LNCS 4631, p2.

I'm particularly interested in this choice of which side a human uses, because that seems like something that would be very hard to do automatically, and you don't want somebody to be able to do that automatically.

Reply: No, you very specifically don't want to make that automatic.

Bruce Christianson: As you say, there's a 50% chance of getting caught every time you do that. That seems like a nice property.

Reply: Yes. I think a nice feature of the two-sided scheme I suggested is this kind of fundamental symmetry between the two sides, so there shouldn't be a psychological bias in which choice, that there was to some extent with David Chaum's original scheme, because it involved visual crypto, and two transparent sheets overlaid, and so on, so there was definitely an upper side and a lower side, and one of the concerns there was whether there would be a distinct bias in voter choice between those two components. Hopefully with this scheme there will be less of a bias.

James Heather: Surely there's a bias whichever side is face up.

Reply: I don't quite know how you solve that, how we get it to print off vertically.

Michael Roe: Physical designs for voting schemes, it's like when we were doing lottery tickets, or raffle tickets, it gets printed and then tumbles down a chute, but hasn't had a chance to turn over a random number of times.

Reply: Yes, maybe there are things like that you can do.

Bruce Christianson: And it's possible that something like this could apply to lotteries and pools as well.

Would your system extend to not first past the post systems, to things like single transferable vote, and so forth?

Reply: Yes it does, certainly STV. You need a full permutation of a candidate list, so you can do that. If you're doing subset choices, that's slightly tricky because of the kind of problem that Matt was saying, you will be able to see how you selected, and that's difficult to avoid unless you have separate onions for each candidate.

Bruce Christianson: Or you could have random permutations between the two columns, you can have a yes column and a no column, but it's randomly swapped.

Reply: I guess you could do something like that, yes. Certainly you can do single transferable votes, and there are additional tricks: for single transferable vote, it's not a single cross, it's a list, a vector of values, and you can actually send these through the mix separately. One of the coercion problems with STV, particularly if you've got a large number of candidates, is that the low order values can be used and identified by a coercer, but if you send these through the mixer separately, you split them up, and so you avoid those kind of things. But yes, it does generalise.

Towards a Secure Application-Semantic Aware Policy Enforcement Architecture

Srijith K. Nair, Bruno Crispo, and Andrew S. Tanenbaum

Dept. of Computer Science, Faculty of Sciences, Vrije Universiteit,
1081 HV Amsterdam, The Netherlands

Abstract. Even though policy enforcement has been studied from different angles including notation, negotiation and enforcement, the development of an application-semantic aware enforcement architecture remains an open problem. In this paper we present and discuss the design of such an architecture.

1 Introduction

As networked and grid computing and web service architectures are gaining acceptance, computer systems are being transformed from standalone systems into a shared and more open environment. Policies defining the limits and working conditions of various parts of such a system constitute a pseudo-contract based environment of operation. Enforcement of these policies in a proper manner plays a crucial role in preserving the trust and integrity of the system.

Even though a lot of research has been carried out in the area of policy enforcement at various levels of abstraction, it still remains an open problem. Most of the policy enforcement approaches have been geared towards enforcement of users polices on a time-sharing machine administered by a central authority [1]. With the advent of cheap secure hardware [2] and technologies like secure booting [3] that are cheap enough to be deployed on individual user machines, we need to re-evaluate existing approaches towards policy enforcement. In this position paper we plan to describe briefly our contribution towards solving some of the issues of the policy enforcement problem.

In Section 2 we give examples of applications whose policies are hard or impossible to enforce with current solutions. Architecture for Zodac, a new secure policy enforcement system is presented in Section 3 and the security considerations of the system are discussed in Section 4. We then provide a brief overview of previous work in Section 5 and conclude in Section 6.

2 Policy Enforcement Examples

Policy enforcement is a critical part of any secure system. In this section we provide couple of scenarios and example applications whose functionality and integrity depends a lot on the presence of such a system.

Mobile Agents – Companies are opening up their services to allow third party generated code (agents) to run on their hosting platform and query their back-end databases [4]. Such mobile agents raise two main security issues - first, the safety of the platform running these untrusted agents and second, protecting the integrity and confidentiality of the data and code of the agent. When a mobile agent is sent off to a remote host, it could be associated with certain policies that define its working constraint. For example, an auction agent could be allowed to share the personal details of the owner, but not the highest bid it is allowed to place. The agent owner needs to be certain that the policies are enforced in an environment that he does not own or control. Similarly, the remote host will also be constrained by policies that define the limits of the service that it can provide. These policies too need to be enforced.

Emails – Corporate emails can be associated with various policies like Do not forward, Forward only within the Accounting Department, To be accessed within protected/safe environment only, *etc.* It is paramount that these policies are enforced in a proper manner, without relying on the judgment or goodwill of the recipient.

Digital Content Usage – To exploit the emerging market for digital media (music, movies, e-book, *etc.*) the content owners may decide to associate various kinds of policies with the content they distribute. For example, a digital audio clip could be associated with the policy that unless a payment has been made, only 30 seconds of it can be played or that it can be played only 5 times. The content owners need to be convinced that these policies will be enforced in the users end machine.

Privacy Data Protection – When a user shares his data with an external party, he may wish to associate a policy governing the usage of the data. For example, he may wish to state that none of the data he has provided should be shared with a third party. More complicated policies could take the form of sharing as long as owner gets paid one cent per transaction or that data should be destroyed after 1 year. The user needs to be certain that the external party has a system in place to enforce these policies. The external party needs to prove that a system exists that can guarantee the enforcement of the policies and then carry out the actual enforcement at its end.

3 Zodac Enforcement Architecture

Our aim in this paper is to describe a general purpose policy enforcement architectural framework, named Zodac, which is generic enough to be used in a wide range of applications, some of which are mentioned in the previous section.

3.1 Components

In order to build a generic framework, we have divided our architecture into components based on the functionality provided by them or on the levels of trust associated with it.

File system – The file system will be implemented as strongly typed with respect to the application. A file that is associated with an email client can only be accessed by a call from the email client and not a text editor. In addition, sticky policies are also associated with every file, defining other usage restrictions.

Application – While no inherent trust is associated with an application, they may have to be rewritten to make use of the layered structure defined in the architecture.

Application Policy Enforcer – Several policies can have attributes closely related to the applications semantics. These cannot be readily interpreted at a very low level. For example a policy related to Play cannot be interpreted directly at the operating system level. Applications calls that relate to these policies have to be intercepted by a higher up layer that can vet through the calls, interpret them and then translate them to systems calls that can be understood at a lower level. The Application Policy Enforcer (APE) does this job.

Base Policy Enforcer – Application semantic independent calls (like file open, read) are common between applications and hence can be intercepted and analysed at a common layer just above the operating system. The Base Policy Enforcer (BPE) is the name we give to this layer.

Policy Evaluator – This is the primary engine that makes decisions on which calls can be allowed as per policy and which of them have to be denied. Since the engine would need application semantic level knowledge to make certain policy decisions, the Policy Evaluator (PE) will be a per-application-type engine. For example, an email PE will be different from a PDF reader PE.

Plugin Extensions – These are extensions provided by trusted parties that provide additional functionality at the APE and the PE level. For example, a music file purchased from Apple iTunes [5] would have policies specific to iTunes that cannot be correctly analysed by a generic APE or PE for a media player. Hence, an extension can be provided by Apple to correctly interpret such policies.

3.2 Control Flow

This subsection describes the flow of control between the various components of the system when an application opens a file for processing. Figure 1 below gives a summary of the flow:

The application opens a file (1) and in turn opens the associated policy (2). Any system /application calls and operations on this file are trapped and forwarded to the APE (3). The APE may forward this call to a plugin extension (3.1) if the functionality requested is provided by a custom extension. The APE (or the extension) checks with the Policy Evaluator (4) (or its extension, 4.1) whether the call should be allowed or not. The evaluator checks the policy (5) and conveys the decision back to the APE or its extension (6/6.1). If the policy allows the call, the APE passes the call to the BPE layer (7), translating the higher level (application semantic) calls to system level calls. The BPE queries the Policy Evaluator again (8) to make

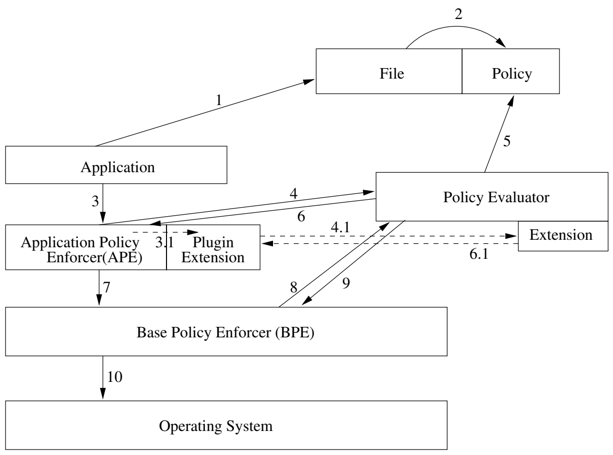


Fig. 1. Proposed enforcement system architecture's control flow

sure that the calls it received is allowed by the policy. The evaluation result is sent back to BPE as denoted in step 9. If allowed, this system call is then sent to the operating system. The resulting reply/message from the OS is then sent back to the APE, which performs the appropriate action.

The additional check in steps 8 and 9 are required to provide a second level of enforcement so that in case a direct system level call is made to the BPE without being routed through the APE, it can be trapped and evaluated.

4 Security Considerations

The application running in user-space is not trusted and hence it can attempt to subvert the enforcement of the policy associated with a file. Our secure framework prevents such security breaches.

We assume the core operating system to be free of exploitable bugs. The operating system accepts only the application calls that have been properly vetted by the APE and the BPE. Jailing [6], reference monitors [7] or other similar implementations could be considered for the actual implementation of this enforcement. A windowing system similar to EROS Trusted Window System [8] or Nitpickers [9] could be used to provide a secure windowing environment to differentiate between different levels of confidentiality of windows.

The APE needs to translate application-calls into low level system-call. The PE (with the help of the extension) evaluates the policy. The policy language should be generic enough to specify most common attributes of usage for different applications and yet extendable enough to be useful for specific kinds of applications. A modification of XACML [10] and XrML [11] could be used as a starting point for such a language. Since the plugin extension systems integrity depends on the plugin provider, a form of trust relationship, maybe in the form of signed codes will have to be established between the provider and the system.

One of the most crucial parts of the enforcement system that is still under investigation is the memory and data management. While a single application can be compartmentalized to prevent data leaks and side channel attacks, the interaction between various applications is a vulnerable area for such attacks. The memory and data management component of the system has to be robust enough to withstand such attacks. We hope to resolve this in Zodac using a memory handle management system wherein the BPE acts as a manager deciding whether to allow or deny request or call for the handle to the applications memory by other applications.

5 Related Work

Most research work that tackle the issue of policy enforcement either concentrate on policy negotiation [12], the decision making process [13], limit themselves to architectures that does not handle application level semantics [14] or consider only the enforcement of policies on machines that are directly under the control of the user [1]. Though a lot of work has been done on access control policy enforcement [15,16], that forms only a part of the system and the other parts like policies based on resource management, semantic level issues *etc.* are left out in such work.

6 Conclusion

In this paper we showed the need for a secure policy enforcement system and how previous research has not provided us one. We then introduced Zodac, a secure policy enforcement system and described the components of the system, detailing the passing of control between them. In the end the security issues associated with Zodac were discussed and couple of possible solutions was presented.

References

1. Multics (May 2006), http://www.multicians.org/
2. Trusted Computing Group (May 2006), http://www.trustedcomputinggroup.org/
3. Arbaugh, W.A., Farber, D.J., Smith, J.M.: A Secure and Reliable Bootstrap Architecture. In: IEEE Security and Privacy Conference, May 1997, pp. 65–71 (1997)
4. Alexa Web Search Platform (May 2006), http://websearch.alexa.com/welcome.html
5. Apple iTunes (May 2006), http://www.apple.com/itunes/
6. The Jail Subsystem, FreeBSD Architecture Handbook (May 2006), http://www.freebsd.org/doc/en_US.ISO8859-1/books/arch-handbook/jail.html
7. Anderson, J.P.: Computer Security Technology Planning Study. Technical Report ESD-TR-73-51, U.S. Air Force Electronic Systems Division, Deputy for Command and Management Systems, HQ Electronic Systems Division (AFSC), Bedford, Massachusetts, Vol. 2(5869) (October 1972)

8. Shapiro, J.S., Vanderburgh, J., Northup, E., Chizmadia, D.: Design of the EROS Trusted Window System. In: Proceedings of the 13th USENIX Security Symposium, pp. 165–178 (2004)
9. Feske, N., Helmuth, C.: A Nitpicker's guide to a minimal-complexity secure GUI. In: 21st Annual Computer Security Applications Conference (ACSAC 2005), Tucson, Arizona, USA (2005)
10. Extensible Access Control Markup Language (XACML). OASIS standards, `http://www.oasis-open.org/specs/index.php#xacmlv1.0`
11. eXtensible Rights Markup Language (XrML) (May 2006), `http://www.xrml.org`
12. Mobach, D.G.A., Overeinder, B.J., Brazier, F.M.T., Dignum, F.P.M.: A Two-tiered Model of Negotiation Based on Web Service Agreements. In: Proceedings of the Third European Workshop on Multi-Agent Systems (EUMAS 2005) (December 2005)
13. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized Trust Management. In: IEEE Symposium on Security and Privacy, Oakland, CA (May 1996)
14. Erlingsson, U., Schneider, F.B.: SASI Enforcement of Security Policies: A Retrospective. In: Proceedings New Security Paradigms Workshop, Ontario (September 1999)
15. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-Based Access Control Models. IEEE Computer 29.2, 38–47 (1996)
16. Graubert, R.: On the Need for a Third Form of Access Control. In: Proceedings of the 12th National Computer Security Conference, October 1989, pp. 296–304 (1989)

Towards a Secure Application-Semantic Aware Policy Enforcement Architecture
(Transcript of Discussion)

Srijith K. Nair

Vrije Universiteit

Matt Blaze: How do you stop me from photographing the screen that displays my mail message?

Reply: The analogue hole is always a problem unless congress does something about it. You could play the music, put a microphone in front of the speaker, and record it; that's always going to be a problem which I don't think it's technically feasible to solve.

Bruce Christianson: You're relying on the fact that most people won't, or that the quality will be sufficiently degraded.

Reply: Yes, that the quality will be sufficiently degraded, but also the kind of system we are trying to build is basically for, like if you are employed in an organisation and all the hardware has been lent to you by that company, and you want to implement a policy on that set of machines, how do you actually go about doing it? A typical example is email.

Matt Blaze: But the email example is different from the music copying example, even though in principle it's digital rights management, because the scale of an email message is vastly different from the scale of all of the nice sounding notes of a piece of music.

Reply: The problem with email is that you can actually have a policy which says, don't forward it to ABC, but you can forward to somebody else; DRM is usually binary, you play or don't play, so this could be a bit more than binary decision-making.

Matt Blaze: Yes, I mean, what I want to do is tag the message with, don't use this message in any way that I will regret.

Reply: Yes, exactly, whistleblower.

Bruce Christianson: There's a fine line because if you're not allowed to forward it, but you can reply to it, and you can circulate the reply and the original message is included in the reply. . .

Reply: That's true, so you have to make a search of how much of the data is actually in the new message. I mean, you could always do a reply, and then change the "to" address, so the thing is, if you have a base policy enforcer, you can never catch it, because at that level it's just binary data, but if it's at application policy enforcer level you can actually tell the difference between whether the "to" has been changed or not.

Bruce Christianson: So the relation between the plugins in the next layer down is, you've got to think quite carefully about it haven't you. OK. That's why you have those two extra calls.

Reply: Yes, that's right.

Mike Bond: I think if I had one high level concern it would be trying to add to a policy enforcement architecture some access control which decides who gets the benefit of policy enforcement. If you have policy enforcement for everybody, then you're going to find that your computer fills up with different applications with different policies, which either conflict with each other, or conflict with your own wishes. To give a simple example, you have a text editor which installs itself and decides to show adverts, but says that only I am allowed to read text files, and then you're stuck with that, because it's enforced its policy on your files.

Reply: Yes but if you looked at the requirements which I had before, one of the things which I wanted to have is a policy that was actually fair to all involved parties. So what happens is, if you want to open a text file, which can be opened only at the text editor, you should be told upfront that this is what's going to happen; if you want to read a mail which says "don't forward", you should be told, OK, you're going to read a mail which is "don't forward" tagged, so do you want to read it or not.

Mike Bond: Yes, what I'm saying is that as you bring in more and more units to these systems, there are decisions that you may not easily be able to go back on. OK, in the case where you've got a steady application basis, a corporation, then I guess it's not a problem.

Reply: You mean the choice of applications, like I could be opening a file in A application, or a B application, and if A sends me a lot of ads, I can switch over to B?

Mike Bond: What I'm saying is, if both A and B have a policy saying, I'm the only person who can read this sort of file.

Reply: Oh, we still want that decision to be able to be made by the user. There is still a fine line between what a user should or should not be allowed to do, I would say the user should be able to choose what application can open a file. So if I'm fed up with two applications, you can get somebody else to write an application for you, or somebody else will feel, OK, this is a good opportunity for me to write that application.

Tuomas Aura: But then you can write your own application that prints the file.

Reply: Exactly, and that's exactly the point we are trying to prevent, because your application is not trusted at any point, so if you can actually catch it at its system level calls, irrespective of what application it is, you should be able to prevent that. That's the biggest problem we are facing.

Bruce Christianson: Your argument is that you're enforcing these semantics, not by controlling which applications run, but by controlling what they do?

Reply: Yes.

Tage Stabell-Kulø: To me enforcement and fair is a contradiction in terms, because enforcement means that you decide something over me, while fair means that I decide what I want to do.

Reply: Yes, but it also brings in the question, who owns the data, which is a big question because if I send an email to you, who does the email belong to?

Tage Stabell-Kulø: It belongs to me.

Reply: But I sent it.

Tage Stabell-Kulø: Yes, you're right. But, if you give something to me, it becomes mine, I'm sorry, this is the way it is.

Bruce Christianson: Your statement is very useful because it's highly controversial. [Laughter] The idea that because I send you something that's copyrighted, I'm relinquishing all my rights over it, is perhaps naïve.

Bruno Crispo: Suppose that you have this service provider who collects personal data, it's your data, but they claim that they don't forward to any third party. In the commercial world, I give you my credit card details, the credit card is not yours, you just use it, but it's still mine, so it's not clear what is the ownership of an email.

Reply: So if I have a mobile agent and I send it to you, that mobile agent is yours, to do what you like.

Bruno Crispo: What belongs to you is the space that it takes, not the information it contains. I think as a general rule, when computer scientists act as lawyers, run for cover. The difference between the copyright of some information, and information itself, is totally a different issue, and now we're hearing lawyer talk, so we should run for cover. Copyright is a legal construct. But if I ask you in advance, look I am sending you something that belongs to me, then you can agree to give some of your hardware, some of your storage or not, accepting that actually you're hosting some piece of information that is not yours.

Tage Stabell-Kulø: So you send me an email, it lands on my desk, and you say that you can ensure that it still belongs to you?

Bruno Crispo: Well for the mail, not.

Kenny Paterson: Let me give you an example, Google mail is very popular, I set-up Google mail, Google has the right to trawl through all of my email, and through certain advertising on the website to try to sell me services, I'm very happy because I get a very large storage area, but I would rather they didn't use that email then to then market other services to me. At the moment that's all done through my trust in the Google brand and wouldn't it be nice if we could have a system which guarantees that.

Tage Stabell-Kulø: Can we trust Google to be careful with our data?

Reply: No, you can't, because there was a recent case where somebody was told to hand over all the data of his mail, even the deleted mail.

Bruno Crispo: Because there is also the owner, so even if you don't want to then you may be forced to hand it over.

Bruce Christianson: Sure, but suppose you are providing the service, and you wish to protect yourself against malicious accusations that you're forwarding mails that are not the mail that was sent to you, and so it's very useful to be able to defend yourself by saying, well I cannot do that.

Tage Stabell-Kulø: Yes, and here is the crux of the issue. Do you know of any other way of protecting information that you do not physically control other than encryption?

Tuomas Aura: Well the email's on your hard drive, but if I sent an email, it is mine. Your computers have access to it, you can read it, but you are not allowed to publish it in a newspaper.

Tyler Moore: But if I send something to you, and it's a Google email account, then Google serves as a third party which also gets this access to communication, that is the part that you want to regulate, what exactly they can do. So we're dealing with different users, we're storing this information, so you have different policies.

Tage Stabell-Kulø: Yes, at the moment I'm getting to feel that legal contracts and copyrights get things much more murky.

Chris Mitchell: It seems to me this whole conversation is invalidated by something that you said earlier on, that you are aiming at a corporate environment where the hardware doesn't belong to me even though, you mentioned, it's on my desk. In which case, this conversation doesn't apply. I suspect this is where all the money is, all these solutions are not for the home user, because we're not going to be running a policy enforcement layer on our own PCs, well certainly I don't suppose I will be. So, I think there's a great danger in getting very worked up about what the corporates do, and trying to apply it to the home user, I think this is where a lot of heat has been generated without necessarily a lot of light.

Mike Bond: I would counter-argue that corporate users have already got perfectly adequate systems, I think IRM[1] for instance, with Microsoft Office worked great in corporate environments, do they really need more?

[1] See `http://office.microsoft.com/en-us/help/HA101003661033.aspx`

Chris Mitchell: Maybe that's true, but I think they do have already difficulty in managing security because things are so much more distributed. This notebook might be part of the corporate environment, they may have bought it, and they may have installed things on it, but it's not within their firewall anymore, they've lost the little control of what goes in and out of this machine.

Tuomas Aura: Can I ask just a purely technical question [Laughter]. If you have communications like, do not print, or, do not forward, on the data, do not email, and you say you allow any application to process it. Now cutting and pasting data from one document to another, that should infect the other document with the same access controls.

Reply: Well, it's a problem of tainting. Your X-Window environment should know the policy differences, and there are ordinary environments which actually take care of it. You have a high priority window which cannot cut and paste in the low priority window.

Tuomas Aura: These windows are now applications that are trusted, but you say, I'm going to write my own applications. If I write my own applications where I can read emails, I can now take, and I can maybe draw paintings, and write books, and so on, so I accidentally cut and paste from one email that says "do not forward, do not print, do not do anything", I cut and paste a section to the book I was just writing.

Reply: Now the book is tainted, yes, but the question is, does the application actually provide the whole X-Window environment, because there is a difference between an application, and what is rendering the data onto your screen.

Tuomas Aura: I can write my own applications, I can write an application that can process any kind of data, it doesn't need to provide any windows, it just has the capability of reading one file that has access controls on it, that I receive by email, and writing another file that I created myself. Now if I can cut and paste anything from the file, with a policy on it, to this document that I have created myself, will my document be tainted?

Reply: It will be tainted.

Tuomas Aura: So do you see what happens, you will have a high watermark system where suddenly, I can never publish this book because I can't print it anymore. Because I accidentally cut and paste something from email to that file.

Reply: Yes, but that accident is always a point where you can actually leak information. Do you want to err on the safe side, or on the sloppy side?

Tuomas Aura: But that's the way we use computer systems, we process data with different tools, and we combine them, and we use pieces of data from here and there, cut and paste is one way of doing that, attaching objects is another way. If there is a policy "do not print", that means anything that is tainted with this data will have a "do not print" on it, eventually everything on my hard disk will have “do not print” on it. Unless I’m very careful not to taint other documents.

Bruno Crispo: But you cannot transfer from one application because it doesn’t succeed to transfer the data.

Bruce Christianson: The purpose of process isolation is to stop you from doing that.

Tuomas Aura: But, so you’re saying that there’s one process for each document, and you can never pool information between two processes. But both processes are running applications that I have written, so they both cooperate as much as possible.

Reply: Yes, but if you’re sandboxed, you can do process isolation independent of the processes.

Tuomas Aura: So this means there is no free communication between any two processes.

Bruno Crispo: Yes, because otherwise you break the policy.

Bruce Christianson: Any IPC must go through a system call that is subject to checking against the policy.

Reply: This argument is, the policy doesn’t say “don’t copy”, the policy says “don’t print”, so in that case, you’re right, the policy taints the next file, but the problem is, do you want to allow that, or not allow that.

Tuomas Aura: It’s not a question of me, it’s the person who sends the email, isn’t it, who sends the policy.

Bruce Christianson: Perhaps he sent you an email saying “don’t publish”.

Tuomas Aura: Is the user allowed to do anything to recover from this?

Michael Roe: Isn’t this just a mandatory access control? You’ve got this classified document, and you are writing a book that’s currently unclassified at a compartment node at my workstation, if you were to cut and paste a paragraph from the classified document you can do two things, you can either upgrade your book so it becomes a classified book, or you can not cut and paste, and the workstation enforces that.

Bruno Crispo: Yes, but it’s not mandatory in the sense that here every user has his own policy for his own documents. There is not a system that actually can force a mandatory access control here, you have a policy attached to a document, it’s not exactly the same. But it is true that this goes really against what people are used to nowadays because, for example, you have a piece of a Word file, and you can go from Powerpoint to Entourage to Office, essentially there is no compartmentalisation of any software applications.

Tuomas Aura: But in these kind of emails you won't have just one compartment, you have as many as there are emails in your inbox.

Bruce Christianson: Yes, yes.

Tuomas Aura: And this makes things really complex, because there are...

Reply: No it doesn't.

Bruce Christianson: No, it's only since the complete lack of self control that arrived with personal computing that we've had this idea that processes are expensive and managing a lot of them is unnecessarily difficult. This is more of a return to the way things worked in the mainframe world.

Tuomas Aura: But you're managing many different security policies.

Bruce Christianson: Where this gets really interesting is exactly where you're composing policies. For example, if I have a project team that's working for a company that are sub-contractors to another company on this project, but the company's bidding against them on a different project, and I want to be sure that they're complying with the security policy for this project, and they're not under management pressure going to, for example, cut corners on testing, or do something that would make us look bad in the other contract that we're bidding for. Here they've got to comply both with their own company's policies, and with the policy of the project in terms of the tender we bid for, now that seems to me to be something where an architecture like this does have something to offer, and it's precisely because you are composing policies. You can't do that in a flat environment.

Tuomas Aura: So that means every time I cut and paste, or any time I move information from one document process to another, I need to read through this complex security policy to check what kind of restrictions there are in place?

Bruce Christianson: You don't have to do it yourself, there's software that does it for you.

Tuomas Aura: Sure, software does it, then I accidentally...

Bruce Christianson: Well if you can do it accidentally, your software's not as friendly as you thought.

Matt Johnson: Can I ask you where you're envisaging all these processes running, is this on a Unix style system where it's all running on a remote server, or are you talking about also having it running on the local machine?

Reply: It's actually much easier to enforce on a local machine.

Matt Johnson: Well can you explain that, because if you've got a local process that's got access to a file, and has all the necessary information to do whatever decryption it needs to get at the data, then why can't I write my own code, possibly by reverse engineering the processes running on my local machine, and get at the data. It seems to be the same problem with encrypted DVDs, that if everything's got to know the key to ever get at the data, then sooner or later somebody's going to leak the key.

Bruno Crispo: Be careful, because the assumption here is the hardware is trusted and the operating system is trusted, so it's limited.

Reply: You see the thing is, first of all you find that you can write your own application, and that you can have access, if you are a user yourself, and you can allow a particular application to have access to the data, the question is, what can you do with the data, it's there in your memory, but what else can you do with it.

Matt Johnson: OK, so this could be some sort of tamper proof computer, so that I can't look at things sector by sector.

Reply: Yes, why not, I mean, it's already there, because when you do process isolation you have all this hardware base support already available to you.

Bruno Crispo: You need also a trusted operating system, because otherwise you're in trouble as well.

Matt Johnson: Well do you need a trusted operating system, so that you can't just rip the hardware out and go and take it somewhere else.

Reply: No. The actual implementation could have things to do with decryption, so even if you take all the hard disk to another machine, all you will have is encrypted data. The actual technical implementation is still being worked on, so how you actually enforce it, is still a problem.

Matt Johnson: But if my machine needs to be able to decrypt what's on the hard drive when it boots up, then my machine needs to know what the key is, and if my machine knows what the key is, then I can get that key out of my machine.

Reply: An example would be TPM. I mean, a TPM has a hardware-based key which you can't get from outside.

Matt Johnson: OK, so you need tamper resistant hardware?

Reply: Yes. You basically build on top of hardware which is processed.

Matt Johnson: You can never email one of these documents to somebody who's using a normal PC at the moment.

Bruno Crispo: Again one of the assumptions is that we don't care about backwards compatability, yes, that's true. Otherwise, it's no way. The world is already difficult enough as it is.

Phish and Chips

Traditional and New Recipes for Attacking EMV

Ben Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Steven Murdoch, Ross Anderson, and Ron Rivest

Computer Laboratory, University of Cambridge

Abstract. This paper surveys existing and new security issues affecting the EMV electronic payments protocol. We first introduce a new price/effort point for the cost of deploying eavesdropping and relay attacks – a microcontroller-based interceptor costing less than \$100. We look next at EMV protocol failures in the back-end security API, where we describe two new attacks based on chosen-plaintext CBC weaknesses, and on key separation failues. We then consider future modes of attack, specifically looking at combining the phenomenon of *phishing* (sending unsolicited messages by email, post or phone to trick users into divulging their account details) with chip card sabotage. Our proposed attacks exploit covert channels through the payments network to allow sabotaged cards to signal back their PINS. We hope these new recipes will enliven the debate about the pros and cons of Chip and PIN at both technical and commercial levels.

1 Introduction

The EMV[1] protocol suite – the technology underlying "Chip and PIN" – has now existed in one form or another for over ten years, though it has only been deployed in Europe for less than two years. Over this period there have been plenty of hypothetical attacks and fixes to the protocol in turn, yet it is only since deployment that there has been enough clarity to fully explore possible weaknesses both at a design and implementation level.

In this paper, we look at the big picture of EMV, exploring the feasability of attacks exploiting the fundamental shortcoming of smartcard-based systems – lack of a trusted user interface. We then look at technical errors in the detail, specifically at how the bank's security API to receive and process messages from chip cards implements the required functionality; we show that several vendors have tripped up here. Finally we consider brand new attack modes that may become important in the future (once the easier vulnerabilities have been counteracted), specifically looking at combining technical sabotage attacks with the ever-problematic phising phenomenon.

We hope that this whistle-stop tour shows that whilst EMV is undeniably a robust and secure payment protocol at heart, there is so much matter and

[1] EMV is named after the original contributing corporations: Europay, Mastercard and Visa.

complexity around the edges to get wrong that there will be plenty to keep the criminals fed and watered in the future; we look forward in particular to *phish and chips*!

Section 2 of this paper describes eavesdropping and relay attacks, section 3 describes API attacks at the back end, and section 4 considers phishing attacks. We conclude in section 5.

2 Eavesdropping and Relay Attacks

While a smartcard is a very convenient form-factor to carry, it lacks a trusted user interface, unlike for instance a mobile phone. This means the PIN cannot be provided directly to the card, so there is the possibility of eavesdropping en-route. Lack of trusted display means there is no way to confirm who you are doing business with, and what amount is being transacted, so it becomes possible to relay the entire data stream to another location. Let's look at these two well-known drawbacks in more detail.

2.1 Eavesdropping POS Terminals

If account and PIN data can be eavesdropped from an EMV transaction at a Point-Of-Sale (POS) terminal, it is easy to make a magnetic stripe card containing that data, for fraudulent use in a foreign country where EMV is not supported. Eavesdropping equipment is already commonplace for unmanned ATMs, usually consisting of a overlay for the card slot and a concealed camera. However there are multiple approaches to eavesdropping POS equipment, each with advantages and drawbacks:

– *Camera and Double-swipe.* The most basic approach requires a collusive merchant. The merchant positions a camera with view of the PIN pad, and secretly swipes the card through his own equipment before inserting it into the genuine device.
– *Hacked Terminal.* A real POS terminal is opened up and additional circuitry/probes are added to monitor the keypad, and record the data from the smartcard.
– *Counterfeit Terminal.* The shell of a genuine POS terminal is used to make a counterfeit, which appears to accept card and PIN, but performs no transaction. Alternatively the counterfeit may pass on the data stream from the smartcard to a hidden genuine terminal, but a physical actuator system to enter the PIN on the real terminal may be required.
– *Terminal Skimmer.* A miniaturised interceptor device can be overlayed on top of the smartcard slot on the POS terminal. From this position it can intercept the smartcard stream (capturing account details), and also the PIN. The PIN can be captured here because most European systems are using the cheaper *Static Data Authentication* EMV cards, which possess no private key, thus the PIN can only be sent in the clear.

The camera and double-swipe approach is definitely workable, but a significant disadvantage is the level of collusion from the merchant required, in order to set up the camera and conceal the second magstripe reader. In addition, a human must go through the video footage study the PINs entered, then correlate with the captured data, which is time consuming and error-prone. Modifying a working terminal requires bypassing of the tamper-resistance, and though this is unlikely to be of a high standard, the attack is still technically very complicated, and requires considerable manual effort for each terminal sabotaged.

The counterfeit terminal approach is appealing, and scales better than a hacked terminal. However the effort required to program a brand-new terminal counterfeit (particularly to drive the receipt printer and LCD display) is substantial. It works well within the business model of giving the customer a free lunch in return for his card and PIN data, but ideally the corrupt merchant would like to mix genuine and counterfeit business over the course of the day. Setting up solenoid actuators so that the PIN can be forwarded is a further complication.

In all, our preferred approach in terms of cost, development time and convenience is to create a skimming device which sits on the smartcard slot, and captures card and PIN data. We created a prototype device using an EZ-USB microcontroller and laptop computer, costing in total comfortably under $100, in development time of approximately one man-month. Our prototype is shown in figure 1.

It can trivially capture and store account details and PINs for SDA cards in large quantities. Such devices exist for smartcard and POS development, but cost more like $2500 per device. A finished device could be slotted onto the terminal in seconds, and removed equally quickly should there be a problem. It is thus easier to deploy in environments with only limited collusion from the merchant, and more deniable than installing complex counterfeit terminal equipment or hidden cameras.

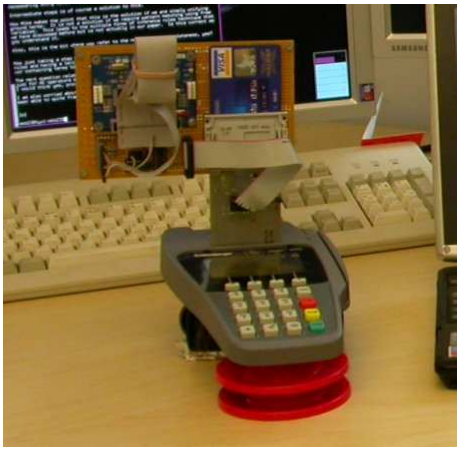


Fig. 1. Prototype POS skimmer

2.2 Smartcard Relay Attacks

Eavesdropping attacks collect account and PIN data for use at a later date, but rely on the magnetic stripe fallback mode of operation, something which might one day be discountinued (though it is unlikely). However if the attackers are well-prepared, they can use the access to the customer's card and PIN in real-time: this is called a *relay attack*.

For example, as you pay for a meal in a dodgy restaurant using your chip card, the waiter's sabotaged reader could simply forward all the traffic from your card wirelessly to an accomplice at a jewellers on the other side of town. The smartcard data stream would go maybe via GPRS to a PDA in the crooks pocket, then to his fake card, and the captured PIN read out via a headphone in his ear. You think you're paying for lunch, but in fact you're buying the crooks a diamond!

This sort of relay attack is a variation on the counterfeit terminal eavesdropping attack, and we imagine the equipment reqired to deploy it would cost less than \$2000, though substantial development and debugging time would be required. We discuss possible countermeasures to the relay attack problem in a forthcoming paper [1].

3 Back-End API Attacks

Back at the bank data centre, a rack of Hardware Security Modules (HSMs) are tasked with providing the back-end support for EMV cards in the field. There are two major roles: *processing authorisation requests and responses*, and *sending secure messages*. An authorisation request or response is simply a MAC over specific transaction data fields, constructed using a specially derived 3DES key shared between HSM and smartcard. A secure message can be thought of as an authenticated script command sent to a card, which usually acts to update some internal variable in the smartcard's non-volatile memory. Secure messages can have encrypted fields, for instance so that a new PIN can be securely sent to the card.

3.1 EMV Secure Messaging in the IBM CCA

IBM's Common Cryptographic Architecture is a popular security API implemented by IBM mainframes and in the 4758. As part of our study of EMV, we looked at the recently-added support for EMV transactions in both the CCA API and the Thales RG7000 series API. We found several vulnerabilities in the support for secure messaging, which are described in detail in a forthcoming paper [2]. These attacks are significant because they show that the EMV protocol has not mitigated the risks of abuse by bank programmers at operations centres, and insider attack there can rapidly undermine the system.

We now briefly describe the attack on the `Secure_Messaging_For_Keys` command of the CCA, which allows us to extract secret keys (and PIN updates) being sent to a smartcard, and inject our own keys and messages without authorisation. The CCA command `Secure_Messaging_For_Keys` is basically a special kind of key export. It takes a key stored locally on an HSM, decrypts it, then formats it up as part of a *secure message*. This secure message format is specified by template input arguments to the command – consisting of a template and and offset at which to insert the encrypted data. The command then re-encrypts the message under a specially derived key shared between the HSM and the destination smartcard. Finally, a separate command `MAC_Generate` is used to create an authentication code over the whole message. Here is the `Secure_Messaging_For_Keys` call in detail:

$$\mathit{template},\ \ \mathit{offset}\ \ , \{K_1\}_{KM/T}, \{K_2\}_{KM/SMSG} \longrightarrow \{\mathit{template}[K_1 : \ \ \mathit{offset}\ \ ]\}_{K_2}$$

– *template*: the message template, a byte-string to be used in preparing the plaintext.
– *offset* : the offset within *template* where the key material should be placed.
– $\{K_1\}_{KM/T}$: K_1 is the payload – a key to export to the smartcard. KM/T represents an encryption key used to store the payload key locally.
– $\{K_2\}_{KM/SMSG}$: K_2 is the key shared between HSM and EMV smartcard. This is used to encrypt the confidential data within the secure message.
– $\mathit{template}[K_1 : \ \ \mathit{offset}\ \ ]$: represents the template plaintext *template* interpolated with key material K_1 at offset *offset* .
– $\{\mathit{template}[K_1 : \ \ \mathit{offset}\ \ ]\}_{K_2}$: the finished result – an encrypted secure message consisting of template with $K1$ interpolated, all encrypted under $K2$.

3.2 Construction of an Encryption Oracle

Our injection and extraction attacks work by gaining access to an encryption oracle. We first note that the CBC mode used in `Secure_Messaging_For_Keys` has an unfortunate malleability property: a ciphertext can be truncated to create a ciphertext of an identically truncated plaintext – so long as the truncation is block-aligned. Thus, we can thus construct an encryption oracle for an arbitrary input message m as follows:

$\mathsf{EncryptionOracle}\Big(\mathit{plaintext}, \{K_2\}_{KM/SMSG}\Big)$:

1. create a template *template* by extending *plaintext* by a single block, *e.g.* the 0-block.
2. set the *offset* to $|\mathit{plaintext}|$, which is effectively the beginning of the 0-block just added.
3. perform the call to `Secure_Messaging_For_Keys` using any available exportable key $\{K_1\}_{KM/T}$:

$$\mathit{plaintext}||\text{“00000000”}, |\mathit{plaintext}|, \{K_1\}_{KM/T}, \{K_2\}_{KM/SMSG} \longrightarrow c$$

the HSM will fill in the last block *template* (as indicated by *offset*) with K_1, leaving the entire *plaintext* component of *template* untouched.

4. consider the first $|plaintext|$ blocks of c, effectively discarding the last block. This truncated value is simply $\{plaintext\}_{K_2}$, our desired result.

This very straightforward observation undermines any security merits of the template-fill-in operation of the HSM – the programmer might as well be able to use the special wrapping key shared between HSM and card in a conventional `Data_Encrypt` command.

3.3 Extracting Keys

Such message injection can compromise the operation of particular cards actively, for instance by constructing a message containing a known PIN for the card. However active attacks at a bank data centre carry a significant risk of revealing the attacker's location, so retrieval of communications keys or PINs without affecting card state is far more dangerous.

We now show how to expand the above oracle into a partial-key dictionary attack mechanism: using this approach, we can rapidly extract the key from any encrypted data field in a secure message, one byte at a time. In our explanation, we use `[ ... ]` to denote hex notation of a single 8 byte block. Here is the algorithm:

$\mathsf{ExtractKey}\Big(\{\mathsf{K}_1\}_{KM/\mathsf{T}}\Big)$:

1. prepare 256 plaintext blocks of the form `[0000 0000 0000 00yy]` where `00` $\leq$ `yy` $\leq$ `ff`.
2. use EncryptionOracle on all of 256 plaintext blocks to generate a dictionary of 256 ciphertexts indexed by the ciphertext: $\{\forall yy, \texttt{00} \leq yy \leq \texttt{ff} : (c, yy)\}$.
3. given any secure messaging key $\{K_2\}_{KM/SMSG}$, make an API call as follows:

   $$\texttt{[0000 0000 0000 0000]},\quad offset\ = 7, \{K_1\}_{KM/T}, \{K_2\}_{KM/SMSG} \longrightarrow c$$

4. compare c against the dictionary of ciphertext-indexed bytes. The match yields the first byte of the key, call it `aa`.
5. in order to discover the next byte of the key, repeat the process with a dictionary built from 256 plaintext blocks of the form `0x000000000000aayy`, with an *offset* of 6. This will yield the 2nd byte `bb` of K_1. By continually shifting the key over by one block, we can extract the entire key, one byte at a time.

For a k-byte key, it takes $257k$ queries to extract the whole key: 256 to build up each dictionary, and one more query to identify the specific key byte. Thus a DES key can be extracted in 2056 queries, while a two-key 3DES key can be extracted in 5112 queries. With such an attack, a key update message between bank and card could be eavesdropped, and then a cloned chip card produced, or PINs could be discovered at will. It is interesting to see that while there is nothing wrong with the concept of a secure message in the EMV standard, the flexibility and extensibility requirements of the protocl have made it difficult to implement in an API. It seems IBM chose to make a general-purpose API command, which supported arbitrary secure messages, but unfortunately was also open to abuse.

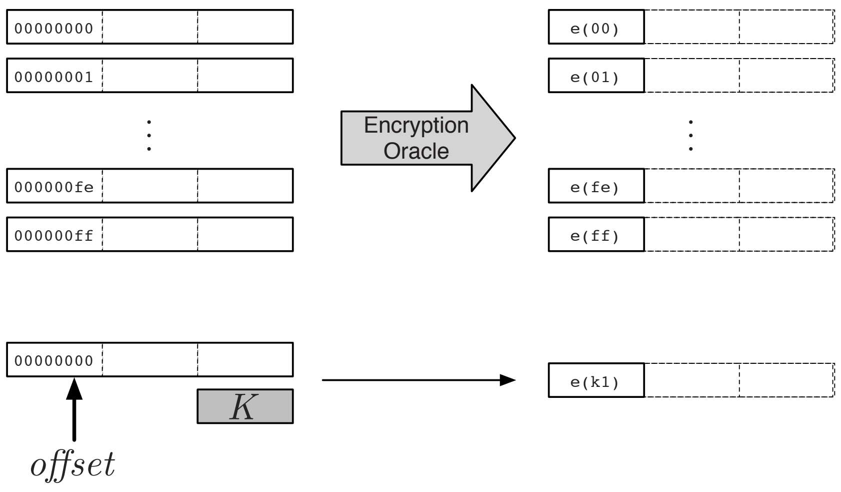


Fig. 2. In the key-shifting attack, a 256-element dictionary is built up for each byte of the key that we want to check

4 Phishing with Chips

Sections 2 and 3 show that EMV users must brace themselves for live deployment of upgraded skimming attacks, as well as inevitable implementation issues with the protocols coming out of the woodwork. But what of blue-sky future attacks, assuming practical measures are found to ameliorate the problems of eavesdropping and back-end attack?

The term *phishing* may have arisen in an internet context, particularly to capture online banking details, but conventional payment systems have been subject to exactly the same social-engineering tricks for some time – by both post and telephone. These include postal redirection scams, unauthorised re-sale (and re-pricing) of another's service, brazenly phoning up and asking for the customer PIN or CVV, and extension of fraud opportunity by faking card cancellation or fraud prevention calls to a customer. But now consider the introduction of chip cards, which has primed the customer to be receptive to new technology and banking requirements with little explanation. Here is an example scam:

1. Open a dummy bank acount in the name of Bob, stick 100 quid in it. Set the PIN to 0000.
2. Set up a skimming scam and get some account numbers, or buy simply some account numbers wholesale. Assume you now have the card number of the victim, Alice.
3. Prepare a counterfeit card from a fictional credit card company bearing Alice's name, but containing Bob's card details.
4. Send the card to Alice, purporting to be a free offer "GBP2000 of credit, and 100 quid pre-charged. Your initial PIN is 0000. Be sure to change it to a memorable number as soon as possible."

5. Wait for Alice to use her free card. Now phone up and request PIN re-advice.
6. When PIN re-advice arrives, it will come to an address under your control, you now have both Alice's PIN (step 5) and her card number (step 2).

The advantage of this scam is that it requires no ability to counterfeit cards, however if opening an account with false name and address is possible the criminal might be better off simply abusing this card straight away. But it highlights the receptiveness of customers to being sent fresh instructions. Consider now how this scam could be improved if the card sent to a customer was sabotaged:

1. Open several bank accounts in false names, F1...Fn. Stick a small float in each (£50–£100 quid). Alternatively apply for several credit cards.
2. Select a target individual Alice, whose address you know.
3. Create a counterfeit card bearing Alice's name, but with an evil smartcard chip on board running your custom firmware. Tell alice that this card replaces her existing one, which she should cut up.
4. When Alice first uses this card, the evil chip will receive her true PIN from the POS terminal in the clear. It now must signal this back to the bad guys using a covert channel through the POS network.
5. Basically, over the next seven days or so, whenever the evil chip detects it is being used for a low value transaction, it switches its identity to assume one of the false identities F1...Fn.
6. The 14 or so bits of information in the PIN are transmitted through this covert chanel of the choice of false identity, and the timing information about which day and hour the identity is switched to. The card has a rough impression of the passage of time from logging the transaction details.
7. The bad guys use internet banking to watch the transaction logs of accounts F1...Fn and from this receive Alice's PIN. They now make up a magstripe of Alice's card and start raiding her account.

Banks have gone to great effort and expense to prevent chip sabotage during the personalisation process, yet a phishing attack provides a perfect solution and bypass of this security – send the customer a sabotaged card and tell her to throw away her old one. Should the fraudsters desire that Alice's card remains fully operational, they need only acquire a legitimate chip through fraudulent means, and embed it in the smartcard in addition to the sabotaged chip. This real chip can then periodically be called upon, for instance if the sabotaged chip wishes for an expensive purchase to succeed.

The key benefit of this scam over brazen solicitation of a PIN (for instance by phone) is that it uses a route through the valid and existing bank network to add the appearance of validity to the scam. The advantage of using choice of account to charge to as a covert channel is that the criminals can collect the resulting PIN data remotely via the internet. A less intricate approach could be to encode the PIN in transaction data which is written onto the paper receipt upon payment (for instance into the *application preferred label* field, or *transaction certificate*). The fraudsters then simply raid the garbage of their target customer, or discarded receipts from a local store in the target area of the scam.

Given the transaction data could be routed anywhere, the card could in fact contact a foreign mafia-controlled bank, where an insider simply reads out the PIN chosen.

5 Conclusions

EMV is a massive and complex protocol suite. To get a proper understanding of where the real security threats are, we must consider all the protocols together. Some of these protocols are conventional cryptographic protocols, such as the secure messaging standard; some are Security APIs, such as the interface to the EMV-compliant HSM, and the API to the smartcard itself. Thirdly, some are human protocols, rules and procedures for authenticating, and communicating secrets to humans. Finally there is the economic aspect, where there are not protocols as such, but there are conscious design decisions which manipulate the threat surface by affecting the cost of attack.

This paper has shown that every aspect of EMV needs to have attention paid to it if the system is going to have the desired security properties. It can hopefully encourage other designers and analysts to look at the big picture and the context of a specific protocol before making a crucial decision. The European banks have a very tough goal if they are trying to wipe out banking fraud; if they are willing to accept some failure modes, it is just as important that when these cracks do appear, they appear in the right places. The specific examples detailed here indicate that the system may already have *unexpected* failure modes: maybe controlling breakdown of a complex system is an even harder challenge than making one that is totally secure!

Acknowledgements

The author would like to thank George Danezis for useful ideas and discussion in relation to phish and chip scams; the suggestion to encode PIN data in receipts from transactions is due to him.

References

1. Anderson, R., Bond, M.: The Man-in-the-Middle Defence. In: Christianson, B., et al. (eds.) Security Protocols 2006. LNCS, vol. 5087, pp. 149–152. Springer, Heidelberg (2006)
2. Adida, B., Bond, M., Clulow, J., Lin, A., Anderson, R., Rivest, R.: On the Security of the EMV Secure Messaging API. To appear in Security Protocols Workshop (2007)

Phish and Chips
(Transcript of Discussion)

Mike Bond

University of Cambridge

Chris Mitchell: From your paper I got the impression you were implying there was only one API, I think actually different banks use different HSMs, with different APIs.

Reply: Yes, I'm going to talk about one API that we looked at, the API produced by IBM, and some of the issues we found with that. We have found issues also with an API from another manufacturer, Thales, and we've got a much larger paper, which is really rather long and tedious, describing all the different APIs we have looked at, the APIs we don't know about, and what we suspect are the cases with those, and I think there's a link to that towards the end.

We talked to IBM about it, there are ways that they can go to fix this by changing the API, but unfortunately they then start to lose the generic functionality, they lose the ability to make this command general enough to capture a load of different messages, and then you have problems. Every time the EMV specification, or a smartcard manufacturer wants a new command, how are you going to change the firmware of the security module to make sure that that particular message can now be added? It's quite a lot of hard work to change the firmware of these things because they're so deeply embedded in a large security system.

So what I've shown you here is specific to the IBM device, but we are writing a paper called, On the Security of EMV Messaging[1], which considers the larger arena.

Kenny Paterson: A quick question, to get that to work presumably you would need to have access to its encryption oracle, does that place limitations on what type of attack can be done, or who can do it?

Reply: Oh yes, if we look at the big picture, different parts of the protocol have access by different people, so customer and card protocol here has access to the customer and access to the merchant, so pretty much anyone. Stuff here is extremely closely protected, and so you've got all sorts of other layers of security surrounding banks and their data centres, where this runs. In practice, despite a wide variety of security vulnerabilities being found in hardware security modules, they can normally go until the next update cycle, without being fixed, or even several update cycles over a couple of years, in my experience, before they're

[1] In Security Protocols Workshop 2007.

updated, because there are so many other layers of security. Banks simply are not experiencing a practical problem with vast quantities of cash going missing from this sort of attack. Although one bank recently did get in the news for a large PIN theft, and we don't know quite what went on there.

The UK banks have decided that they will issue re-advice of your PIN, so if you forget your PIN, you can be sent a letter which tells you what the PIN you chose was. It doesn't give you a brand new one randomly thought up, but tells you your old PIN. So now you can open a dummy bank account, maybe stick $100 in it, and set the pin to all zeros. You capture the details of some victim Alice, and prepare a counterfeit card, so you need a card printing facility, and you make up a fake card, a fake hologram, but you put Bob's details on, and Alice's name. Then you put this in the post to Alice. In the United States, you get sent unsolicited credit cards all the time, you either make it unsolicited, or you say it's a replacement, and you tell Alice there's a $100 free gift if you start using our credit card, by the way, your initial PIN is all zeros, be sure to change it to a memorable number as soon as possible. So Alice goes away with her brand new card, oh this is good, I'll spend $100. Bob whose actual card it is, and whose authentication mechanisms to the bank, via postal address and passwords, are all set up, then phones the bank and says, "I've forgotten my PIN, could I have re-advice please", and Alice's new PIN will be sent to Bob's address.

Michael Roe: Don't they send you a new random key when you do that?

Reply: No, they don't, not all banks do. Quite a lot of banks in the UK issue PIN re-advice, which is your same PIN that you've forgotten sent back to you.

Michael Roe: Which is a bad idea for the reason that you've just told us.

Chris Mitchell: I guess the stuff at the end is really neat, and it applies to just about any conceivable smartcard based payment system, it's not specifically EMV really?

Reply: Yes, I would suppose not. It is specific to EMV, and specific to the UK, I suppose, in that there are options for where a PIN can be routed. You don't have to send the PIN to the smartcard, you could design an electronic payment system with smartcards that sent the PIN over the phoneline back to the bank.

Chris Mitchell: But then I'd worry that there are other scams?

Reply: But then there are different tricks, yes.

Bruno Crispo: To activate a card don't you need a phone?

Reply: In some systems you do, other systems you don't, it depends. That's very much a practical issue: none of these last attacks presented are practical, they're all trying to prove the concept. I think about one in five UK banks require you to phone to activate, mainly the internet banks. The High Street ones don't tend to bother their customers with that.

Srijith Nair: Actually that's the same question I wanted to ask, isn't there a set of guidelines for all of the EMV countries stating how to issue cards? In the

Netherlands, if you want to activate the card, I think you have to bring the card back to the bank to activate it, or if you are asking for a replacement card, your PIN is actually the same as the previous PIN, so you can't put it as 0000.

Reply: Well if so, the trip-up that's happened is that they called the document "guidelines", because the UK banks aren't following it. The UK banks have decided to prize convenience over everything, and make sure that the transition is really easy for the customer.

Richard Clayton: If you're actually sending bad cards to Alice with evil chips on them, not only has the bank not read the guidelines, Alice hasn't read the guidelines either, so she won't ring up the issuing bank in order to get it validated. It really doesn't matter whether that's the normal practice, since you're sending her a special card, you can put a big thing on the top saying, you do not need to ring up in order to activate this card, we'll send you a magic PIN.

Bruce Christianson: Well if it's really Bob's card, and he hasn't rung up about it yet, she might get noticed.

Reply: Oh well, I mean, one of the drawbacks of these things is that some of these attacks are predicated on being able to open bank accounts in false names. If you can do that, you might as well just get an overdraft and start spending. Talking about what is practical and easiest for criminals is a different matter from talking about what's possible.

Alex Shafarenko: Within the same assumption that you can open a bank account, I think, stop this smartcard, they charge £25.37 to Bob, which would be the communication of the PIN 2537, right, so you don't need to print anything on the receipt, it's accomplished two transactions in one go.

Reply: The smartcard doesn't have much control over the value in the transaction in EMV. The smartcard just MACs the transaction data, it's the terminal which controls how much money is spent, so you would have to have a dodgy terminal as well, which would change the threat model a little bit. To be honest, the first avenue of attack is interception relay, this is the area of contention for chip and PIN, whilst magnetic stripe fallback runs alongside it. About 450,000 point of sale terminals in the UK accept PINs, thus increasing dramatically the number of different places that attackers can skim PINs for use with the old system. Once they can shut down magnetic stripe, then it's not an issue, but that's not going to happen for a while.

Where Next for Formal Methods?

James Heather and Kun Wei

Department of Computing, University of Surrey, Guildford, Surrey GU2 7XH, UK
{j.heather,k.wei}@surrey.ac.uk

Abstract. In this paper we propose a novel approach to the analysis of security protocols, using the process algebra CSP to model such protocols and verifying security properties using a combination of the FDR model checker and the PVS theorem prover. Although FDR and PVS have enjoyed success individually in this domain, each suffers from its own deficiency: the model checker is subject to state space explosion, but superior in finding attacks in a system with finite states; the theorem prover can reason about systems with massive or infinite states spaces, but requires considerable human direction. Using FDR and PVS together makes for a practical and interesting way to attack problems that would remain out of reach for either tool on its own.

1 Introduction

Security protocols are vital for providing secure communication and processing of information across a distributed system. However, designing such protocols is notoriously error-prone, since it is difficult to predict and allow for all the possible interactions between the parties operating on the network running the protocol.

Constructing proofs of correctness by hand can be arduous. Indeed, many convincing hand-constructed 'proofs' of correctness of protocols have been published in the literature only to be found wanting at a later date. Over the past decade, formal methods have been remarkably successful in their application to the analysis of security protocols with the emergence of some powerful verification tools.

There are essentially two approaches: model checking and theorem proving. Under a model-checking approach, a system executing the security protocol is represented as a transition system with finitely many states. The model checker then uses various efficient state exploration techniques to discover whether the system can reach a state representing a security violation. Many different model checkers have been employed in this fashion; for example, FDR [1,2,3] has proved to be an excellent tool for modelling and verifying safety properties such as authentication and confidentiality. One has to be extremely careful when using a model checker for such tasks, however: it is all to easy to allow the state space to become unmanageably large.

The alternative is the theorem-proving approach, in which a system and its properties are described by logical formulae, and the formal proof is established by proving theorems that state that such properties hold in the system. One successful such setup is that of the rank functions theory [4] embedded in the PVS theorem prover, which can tackle authentication properties of protocols running on networks involving arbitrarily many agents and with an arbitrarily large message space. However, even when using semi-automated (interactive) provers such as PVS or Isabelle, it is a large task to validate a complex system. For example, in the project to verify SET [5], a e-commerce protocol, Isabelle presents the user with subgoals that are hundreds of lines long, and diagnosing a failed proof requires meticulous examination of huge formulae.

The model-checking approach is superior in finding attacks in a system with finite states, but subject to the state explosion problem; the theorem-proving approach can reason about systems with massive or infinite states spaces, but does not provide automatic verification. One natural question to ask is whether it is possible to blend the two complementary approaches in an elegant way to avoid the weaknesses of each.

There have been various lines of investigation for creating hybrid systems. For example, Cohen [6] proposes a proof method for analyzing security protocols in which safety properties are proven by ordinary first-order reasoning, and all proof is generated in an automatic verifier, TAPS. Song [7] also proposes an efficient automatic checking algorithm, Athena, which incorporates its own logic and exploits several state space reduction techniques based on an extension of the Strand Spaces Model [8]. Heather [9] develops a tool, RankAnalyser, that makes use of results [10] to construct a rank function and verify a protocol automatically. It is appropriate for verifying networks of arbitrary size, and with arbitrarily many concurrent executions of the protocol.

However, the above tools are designed for analyzing a few specific properties, all of which are safety properties. Liveness properties—deadlock, non-repudiation, denial of service, and so on—have not yet been mastered to the same degree since they must be expressed in a more complex model. We here propose the novel idea of using the process algebra CSP to describe the system executing the security protocol and the security properties to be verified, and construct the proof of correctness by using a combination of FDR and PVS.

The general approach is to start by modelling the (infinite-state) system in the CSP semantics that we have embedded into PVS, and then start to prove the theorems using PVS. In the course of constructing the proofs, we invariably encounter some subgoals involving only finite-state processes. It would take a long time to trace through the states one by one checking for correspondence in PVS, whereas FDR can verify such cases very quickly; therefore, we proceed by building these results into the PVS theory as axioms, and then proving them correct in FDR. In this way, we harness the power of the theorem prover for establishing results about an infinite-state system, whilst retaining the speed and automation of a model-checker for certain appropriate parts of the proof.

Currently, translating between the PVS syntax and the FDR syntax is done manually; however, we are making progress towards a tool to perform this translation automatically.

2 CSP Notation

CSP is an event-orientated language for describing concurrent systems and their interactions. A security protocol is a concurrent system in which a series of messages are exchanged among the various parties involved. CSP is therefore well suited to the modelling and analysis of security protocols.

In CSP, a system can be considered as a process that might be hierarchically composed of many smaller processes. An individual process can be combined with events or other processes by operators such as prefixing, choice, parallel composition, and so on.

Stop is a stable deadlocked process that never performs any events. The process $c \rightarrow P$ behaves like P after performing the event c. A event like c may be compounded; for example, one often-used pattern of events is $c.i.j.m$, consisting of a channel c, a sender i, a receiver j and a message m.

The external choice $P_1 \Box P_2$ may behave either like P_1 or like P_2, depending on what events its environment initially offers it. The traces of internal choice $P_1 \sqcap P_2$ are the same as those of $P_1 \Box P_2$, but the choice in this case is non-deterministic.

The process $P_1 \; |[\, A \,|\, B \,]| \; P_2$ is the process where all events in the intersection of A and B must be synchronized, and other events within A and B can be performed independently by P_1 and P_2 respectively. An interleaving $P_1 \; ||| \; P_2$ executes each part entirely independently and is equivalent to $P_1 \, |[\, \emptyset \,]| \, P_2$.

The process $P \setminus A$ will pass through the same events as P, but events in the set A become be invisible. The renamed process $P[a \leftarrow b]$ means that the event a is completely replaced by b in the process P. In addition, processes may also be described recursively whenever such descriptions are well defined.

A trace is defined to be a sequence of finite events. A refusal set is a set of events from which a process can fail to accept anything no matter how long it is offered; $\mathit{refusals}(P/t)$ is the set of P's refusals after the trace t; then (t, X) is a failure in which X denotes $\mathit{refusals}(P/t)$. If the trace t can make no internal progress, this failure is called a *stable failure*.

Liveness is concerned with behaviour that a process is guaranteed to make available, and can be inferred from stable failures; for example, if, for a fixed trace t, we have $a \notin X$ for all stable failures of P of the form (t, X), then a must be available after P has performed t.

Verification in FDR is done by means of determining whether one process refines another. In the stable failures model, this equates to checking whether the traces and failures of one process are subsets of the traces and failures of the other:

$$P \sqsubseteq_F Q \equiv \mathit{traces}(P) \supseteq \mathit{traces}(Q) \wedge \mathit{failures}(P) \supseteq \mathit{failures}(Q)$$

For the properties we are considering, if P meets the properties we are verifying, then Q also meets them if Q refines P. For a fuller introduction, the reader is referred to [11,12].

3 Case Study

An elegant example to demonstrate the power of the combination of PVS and FDR is the dining philosophers problem. We imagine n philosophers sitting at a table with a bowl of spaghetti in the middle. Between each pair of adjacent philosophers, there is a single fork; and to eat, a philosopher must be holding both of the forks that are beside him. We assume all philosophers pick forks up in the same order—right hand first—and do not put down either fork they have picked up until they have grabbed both. There are various ways in CSP to model this problem, and one of them is as follows:

$$\begin{aligned} PHIL_i =& pickup.i.i \rightarrow pickup.i.i \oplus_n 1 \\ & \rightarrow putdown.i.i \oplus_n 1 \rightarrow putdown.i.i \rightarrow PHIL_i \\ FORK_i =& pickup.i.i \rightarrow putdown.i.i \rightarrow FORK_i \\ & \Box pickup.i \ominus_n 1.i \rightarrow putdown.i \ominus_n 1.i \rightarrow FORK_i \\ COLLEGE =& \Big\|_{i=0}^{n-1} (PandF_i, AP_i) \end{aligned}$$

where '$\oplus_n$' denotes addition modulo n and $PandF_i$ is the combination

$$PHIL_i \,|[\, AF_i \,]|\, FORK_i$$

The alphabet sets used are:

$$\begin{aligned} AF_i &= \alpha PHIL_i \cap \alpha FORK_i \\ AP_i &= (\alpha PHIL_i \cup \alpha FORK_i) \setminus AF_i \end{aligned}$$

Obviously for the dining philosophers problem, the one and only situation causing deadlock is that in which all philosophers hold their right-hand fork simultaneously and wait for their neighbours to put down their forks. There are many modifications one can make to avoid deadlock, one of which results in the asymmetric dining philosophers problem: one philosopher picks up a left-hand fork first.

The basic strategy we adopt is similar to an induction used in [13], where the authors use a hierarchical compression technique in FDR to prove the case with huge numbers of philosophers, but not with arbitrary numbers of philosophers. The key idea is that we can prove that any number greater than one of right-handed pairs of philosophers and forks are equivalent by hiding their internal events and carefully renaming their interface events. The proof starts from the case with $n = 3$ philosophers; then, for the inductive step, we assume that the case of $n = k$ philosophers is deadlock-free, and show that the system remains deadlock-free when the number of philosophers is $n = k + 1$.

Figure 1 shows the dining philosophers network's structure, composed of philosopher/fork pairs. This figure also shows how we deduce deadlock freedom of $k + 1$ philosophers from the case of k philosophers. The real thing we

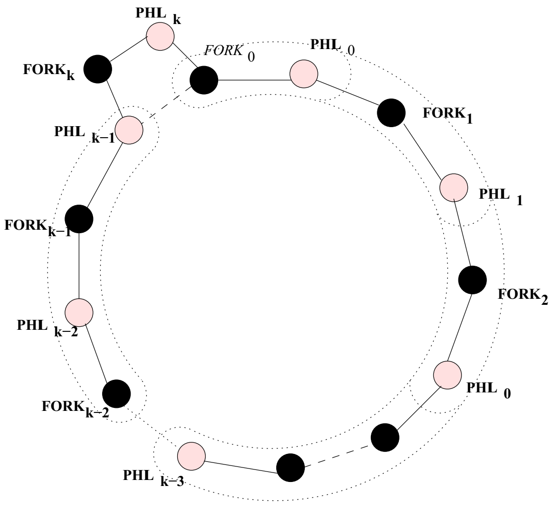


Fig. 1. Inductive structure of dining philosophers

want to achieve behind this step is to prove the equivalence of two processes: k philosophers and $k+1$ philosophers. Of course, it is unnecessary to compare all pairs, and we need to concentrate only on the last two pairs in the circle of figure 1. The key to the induction is that if we hide the internal events of the synchronization of $PandF(k, k-2)$ and $PandF(k, k-1)$, it is equivalent to the synchronization of $PandF(k+1, k-2)$, $PandF(k+1, k-1)$ and $PandF(k+1, k)$ with their internal events hidden and $pickup.k.0$ and $putdown.k.0$ renamed as $pickup.(k-1).0$ and $putdown.(k-1).0$ respectively.

The above key lemma is formally described as following:

$$(PandF(k, k-2) \;|[\, AF_{k-2} \,|\, AF_{k-1} \,]|\; PandF(k, k-1)) \setminus IE_k =$$
$$f((\|_{i=k-2}^{k} \, (PandF(k+1, i), AF_i)) \setminus IE_{k+1})$$

where IE_k and IE_{k+1} denote the sets of internal events and f is a bijective function which renames $pickup.k.0$ and $putdown.k.0$ as $pickup.k-1.0$ and $putdown.k-1.0$ respectively and vice versa.

Although it would be possible to prove this lemma in PVS, it would be in one sense perverse to do so, since it is essentially a very small model-checking exercise. Also proving such a lemma in PVS is rather time-consuming. The natural approach is to establish an axiom for the above lemma and finish the proof in PVS; we transform the script into an FDR script containing an assertion that this lemma holds, and finish the proof using FDR. Using PVS in combination with FDR, then, we can successfully and elegantly prove the asymmetric dining philosophers network with an arbitrary number of philosophers to be deadlock free.

4 Conclusion

Although the example above is not explicitly security-related, we have also found this approach to be highly effective when considering security protocols. For example, we have analyzed and verified the fairness property of the Zhou-Gollmann non-repudiation protocol using a combination of PVS and FDR; this could have been used as the case study, but the dining philosophers example is considerably more transparent, and we considered the digression from the security theme to be a price worth paying for the sake of clarity. The net result of following the PVS/FDR approach is a proof that is automated as far as possible, but that can handle infinite-state systems with minimal effort.

Our aim is to take this work further by automating the translation. We are in the process of developing a tool that can transform PVS scripts into FDR scripts, in order to speed up the process and to avoid introducing unnecessary human error. Ultimately, the procedure will be:

1. model the (infinite-state) system in PVS;
2. use PVS to reduce the proof obligations to finite-state checks;
3. run the translation tool, which will pick up the PVS script and the partially completed proof, and translate the proof obligations into an FDR script containing these obligations as assertions;
4. run FDR on these assertions to complete the proof.

The final stage, we envisage, will involve running a second (far simpler) tool that will run FDR on the resulting script, analyze the results, and insert the proof obligations into the PVS script as axioms for any checks that succeed.

References

1. Roscoe, A.W.: Modelling and verifying key-exchange protocols using CSP and FDR. In: Proceedings of 8th IEEE Computer Security Foundations Workshop (1995)
2. Lowe, G.: Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In: Margaria, T., Steffen, B. (eds.) TACAS 1996. LNCS, vol. 1055, pp. 147–166. Springer, Heidelberg (1996)
3. Ryan, P., Schneider, S., Goldsmith, M., Lowe, G., Roscoe, B.: The Modelling and Analysis of Security Protocols. Addison Wesley, Reading (2000)
4. Dutertre, B., Schneider, S.A.: Embedding CSP in PVS: an application to authentication protocols. In: Gunter, E.L., Felty, A.P. (eds.) TPHOLs 1997. LNCS, vol. 1275. Springer, Heidelberg (1997)
5. Paulson, L.C.: Verifying the SET protocol: Overview. In: Abdallah, A.E., Ryan, P.Y.A., Schneider, S. (eds.) FASec 2002. LNCS, vol. 2629, pp. 4–14. Springer, Heidelberg (2003)
6. Cohen, E.: Taps: A first-order verifier for cryptographic protocols. In: 13th IEEE Computer Security Foundations Workshop — CSFW 2000, Cambridge, UK, July 3-5, pp. 144–158. IEEE Computer Society Press, Los Alamitos (2000)
7. Song, D.X.: Athena: a new efficient checker for security protocol analysis. In: Proceedings of 12th IEEE Computer Security Foundations Workshop (1999)

8. Thayer, J., Herzog, J., Guttman, J.: Strand spaces: Proving security protocols correct. Journal of Computer Security (1999)
9. Heather, J.A.: Oh! ... Is it really you?—Using rank functions to verify authentication protocols. Department of Computer Science, Royal Holloway, University of London (2000)
10. Heather, J.A., Schneider, S.A.: Towards automatic verification of authentication protocols on an unbounded network. Technical Report 00-04. Royal Holloway, University of London (2000)
11. Roscoe, A.W.: The Theory and Practice of Concurrency. Prentice-Hall International, Englewood Cliffs (1998)
12. Schneider, S.A.: Concurrent and real-time systems: the CSP approach. John Wiley & Sons, Chichester (1999)
13. Roscoe, A.W., Gardiner, P.H.B., Goldsmith, M., Hulance, J.R., Jackson, D.M., Scattergood, J.B.: Hierarchical compression for model-checking CSP or how to check 10^{20} dining philosophers for deadlock. In: Brinksma, E., Cleaveland, R., Larsen, K.G., Margaria, T., Steffen, B. (eds.) TACAS 1995. LNCS, vol. 1019, pp. 133–152. Springer, Heidelberg (1995)

Where Next for Formal Methods?

(Transcript of Discussion)

James Heather

University of Surrey

Tuomas Aura: Why do you need to model the medium separately, why not just merge Alice and the medium?

Reply: You could merge them. If you were doing this in the model checker, that's certainly what you would do, because you increase the state space by having the medium done as a separate process, you're right. When you're doing it in PVS you don't have the problem of increasing the state space, so it doesn't really make a lot of difference whether you merge Alice with the medium or not. But yes, you could easily do that.

There's no intruder modelled in there either because what we're going to do is to get Alice to act as the intruder. The purpose of something like an authentication protocol is to protect the two participants from malicious behaviour coming in from a third party. A non-repudiation protocol doesn't work like that, you're trying to protect Alice from malicious action by Bob, and Bob from malicious action by Alice.

The enemy's, the intruder's, actions in the standard model are incorporated into Alice. So we've got no guarantee that she'll play by the protocol rules, but we assume that Bob will. Now this only gives you half a proof, because this only proves that the responder of the protocol gets a fairness guarantee if the initiator plays by the rules, but you could equally move the intruder model to B, and get the same thing the other way round.

Mike Bond: I must just say I'm curious, you really had to prove over a thousand lemmas to get yourself up and running with the system?

Reply: It depends what you mean by lemma, I mean, yes, you prove something and it decomposes and...

Mike Bond: But because you're proving only positive properties of protocols, it doesn't matter if you happen to have forgotten something in that library?

Reply: Yes, that's right, so there might be obvious bits that we just haven't remembered, and some of what we did was to go through lists of CSP laws and prove them regardless, so I don't really mean to suggest that everything that we've proven was necessary to do this work.

Mike Bond: And if that library's deficient, that doesn't damage the correctness of the proofs?

Reply: No, absolutely not.

Bruce Christianson: It may damage the liveness of the proof process though.

Reply: Yes, it just makes it take a bit longer.

Mike Bond: On one of the slides you said, PVS had been chosen for its "Expressive libraries plus wealth of support." Did you find it easy to use?

Reply: It's not something where if you've never used it before you can just sit down and get to grips with it within 20 minutes. Once you are up and running with it, and you know what you're doing, it works quite nicely. It's a bit like moving from Windows to Linux, you might be glad three years later that you did it, but you've got quite a headache getting there.

Mike Bond: Three years? [Laughter]

Peter Ryan: So the reduction to finite state, I wasn't quite clear quite how that was happening, is data independence the kind of trickery you're using in that process?

Reply: Yes, well, it depends on quite what you're trying to prove, but things like the data independence will help you get there. There's no automation of the reduction from infinite state to finite state, but what you can do is construct theorems and lemmas that apply fairly neatly, and fairly quickly, to get you from something that's infinite state to something that's finite state. So, you can prove, for instance, that in certain cases it doesn't matter if a key is reused, and so you can shrink the set down so you've only got a finite number of keys in it, and prove that it doesn't matter that it's only a finite number of keys.

Peter Ryan: Is that a paper proof, or a mechanised proof?

Reply: It's mechanised, but it's not automated. You have to do it in PVS, so it's formally constructed, and rigorous, but it's not automated, it's not until you get to FDR that you get the automated support.

Michael Roe: How automated is proof sharing? The right thing to do, is to have every protocol accompanied by a machine checkable CD Rom, which proves everything to the server. Can you do that for security protocols, every time somebody suggests a protocol, that there comes a CD Rom which has a fully formal machine verifiable proof that it works? The Zhou-Gollmann type protocols would be a good target because of the number of times I have to referee papers proposing variants on that scheme.

Peter Ryan: And then find it's broken.

Ross Anderson: Better still, don't let anybody patent a protocol until they provide 100 gigabytes of machine checkable proof.

Reply: Yes, that's the beauty of using a theorem, that you do then get a machine checkable proof at the end of it. So what you'd get out of this actually is two things, you'd get a machine checkable PVS proof that would have some assumptions built into it, and then you'd get an FDR script that would contain those assumptions, and that FDR script would be checkable as well. Now that doesn't quite get you machine checkability of proof of the whole thing, because the translation...

Michael Roe: Right, that's what I'm worried about.

Mike Bond: Well why should we trust the translation less than we trust FDRs to give us the right answers?

Reply: Well there is that, and then you've got to trust your hardware that it's doing the right FPU operations that you're throwing at it, and things like that, you can go as far as you like.

Michael Roe: I usually trust the verification software, but on the disk that's given to me are two different statements of the theorem: one in PVS and one in FDR. If I just have to trust that these really are the same theorem, then there's a problem. But if there's some tool that churns on and says, that one is just a direct translation of the other...

Reply: Yes, OK, you've got to trust that translation tool to the same extent that you've got to trust the PVS proof checker, and the FDR checker.

Tuomas Aura: The point is that someone else from another university can write another translation checker, and check the same proofs and the same translations, independently.

Reply: Yes, that's right.

Michael Roe: But if there's any kind of human element that can't be verified automatically in the translation, then you've got a problem, for example if there are any handwaving arguments in the translation.

Reply: Yes, that's absolutely right. Getting a formal proof of the translation mechanism would be a massive amount of work, but there's nothing to say that in principle it couldn't be done.

Bruce Christianson: You only need to verify the translation though. You don't need to produce it automatically, you only need to verify it automatically.

Reply: I don't want to be too dogmatic because we haven't finished writing the translation stuff yet, but I don't see why in principle it shouldn't be doable. It might be possible to go a slightly different route and get the model checking effectively to be done within PVS. If you could find an automated way of turning the finite state assertion into 100 Meg of theorem about each state of the system, you then have something that you could put through your PVS proof checker. If all you want is machine verifiability, then you might be able to go down that route. It would be much, much slower than FDR to run the check, because it's doing something it's not really intended for, but it would work, and in principle you could do that. There's nothing that I've talked about that couldn't in principle be proved in PVS, it's just not what it's designed for, it's not designed to let you go through everything state by state. But, yes, it could be done.

Cordial Security Protocol Programming
The Obol Protocol Language

Per Harald Myrvang[1] and Tage Stabell-Kulø[2]

[1] Bodø Graduate School of Business, University College of Bodø, Norway
`permyr@gmail.com`
[2] Department of Computer Science, University of Tromsø, Norway

Abstract. *Obol* is a protocol programming language. The language is domain specific, and has been designed to facilitate error-free implementation of security protocols.

Selecting the primitives of the language is, basically, concerned with determining which issues needs to be visible to the protocol programmer, and which can be left to the runtime without further ado.

The basic abstractions of Obol has been modelled after the ones offered by the BAN logic of authentication. By building on these abstractions Obol makes it less hard to bridge the gap between logical analysis and implementation.

Obol has been designed with the implementation of security protocols in mind, but the language can be used to implement also other types of protocols.

At the core of the design and implementation is pattern-matching machinery enabling the runtime to parse packets as they arrive in order to free the programmer from a wide range of low-level issues know to foster all sorts of implementation difficulties.

1 Introduction

The interest in security protocols stems from the fact that it is impossible to build any non-trivial distributed systems without them. Unfortunately, due to their harsh operating environment, they are notoriously difficult to design [1]. The problem partly stems from the difficulty of establishing a firm foundation for expressing the goals the protocols strives to achieve.

Implementation of these protocols is a different matter all together. There are three main issues: Traditional implementations issues, assuring that security assumptions holds, and difficulties related to cryptography. We will now discuss them in turn.

– In the context of security protocols, any seemingly innocent implementation error can render the whole endeavor useless. This means that implementation efforts must be undertaken with particular rigor. However obvious this might seem to be, there are abundant examples of programming errors leading to all sorts of problems. Also, since there is a substantial degree of freedom in the hands of the implementor, a wide range of implementations could all be correct even though they are incompatible – by using different cryptographic primitives, message representation, naming schemes, error handling, and so on.

– A typical security assumption is that all (honest) participants have synchronized clocks, a task of considerable complexity [2]. This assumption can be used by including a time stamp in messages, to determine when a certificate is deemed invalid, and so on. Another common security assumption is the availability of an infrastructure to support the use of public keys (a PKI). Also this is a task of considerable complexity, see *e.g.* [3]. This type of assumptions are not related to the implementation of protocols themselves, but it would be an advantage if as many such assumptions as possible were visible to those conducting the implementation. Furthermore, using protocols in a computer system implies that it needs to be implemented in some programming language. This makes the compiler (or interpreter) part of the trusted computing base (TCB), together with supporting libraries, the operating system with it's dynamic runtime, and what not [4,5,6]. The alternative, to build all necessary software components (compilers, libraries, runtime, and so on) from scratch, is for most projects not a realistic option. Thus, most implementations must strike a balance between implementation complexity, desire for powerful assumptions, and confidence in the correctness of the implementation when it ready for deployment.
– The last family of issues are the application of cryptographic primitives. Even when the primitives themselves are correctly implemented, applying them correctly is a challenge in itself [7]. These results are not only of academic interest, as some high profile cases demonstrates [8]. Application of cryptographic primitives are hard to get right also at a higher level of abstraction [9].

It is also problematic that, in general, a design "freezes" the world view of the system. The world is dynamic, users' view of the world changes, but a computer system is more often than not a fixpoint around which users invent solutions to make the system do what they need it to do now [10]. In the security domain this is troublesome as the design and implementation of security features is funded on a careful evaluation of how the can be expected to be used. It is an inherit security problem that system can not (easily) be modified in a controlled manner.

The question is then: How can a system that contains protocols change and adapt to new operating environments? In particular, how can a protocol be changed? There can be many reasons for changing a protocol, from emergencies to "just" non-common events such as (temporal) failure of a PKI. In more concrete terms: Because implementing security protocols is a major undertaking it is unlikely that it will be changed unless it is hopelessly outdated, although it would be an advantage not having to wait quite that long. On the other hand, as the computing milieu diversifies it seems reasonable to arrange for flexibility also in the protocol domain. To this end we can only hope for a looser coupling between design and implementation in that it must neither be an Herculean task to alter a protocol that has been deployed nor replace the protocol altogether.

The rest of the article is structured as follows. In Section 2 we discuss what primitives and abstractions a security-protocol programming language should have and support. This discussion is taken forward, in Section 3, where we discuss the design choices underpinning Obol and it's runtime Lobo. Three different implementations are presented in Section 4, while several applications are discussed in Section 5. Section 6 is concerned with related work and other approaches similar to ours. Some conclusions are offered in Section 7.

2 Framework

It is hardly lack of understanding in the scientific and engineering community that is causing problems within the realm of security protocols [9,11,12]. In our view the challenge is to tackle the considerable "gap" between the manner in which protocols are presented, discussed and analysed, and the programming environment that are used to implement them.

All programming constructs are exercises in abstraction. The design choices should ensure that the essential parts of the target problem domain can be expressed and manipulated. That which is irrelevant should be dealt with by the runtime.

Below we will discuss the rationale behind the functional and non-functional requirements placed on Obol.

2.1 Non-functional Requirements

It has been suggested, although for a different type of protocols, that carefully selected abstractions is a possible solution [13]. By constructing a domain-specific language, designers are forced to adhere to established engineering techniques, resulting in language abstractions that prevent obvious mistakes, while still being useful for expressing new ideas and concepts. The first question is whether to offer suitable abstractions as libraries in an existing language (such as Java), or design and implement a language proper. Because we do not believe a system-wide view is the correct one, we can not assume that both sides of the protocol is implemented in the same language. Thus we believe Obol should be a programming language proper, for which several implementations can exist.

Because our language is targeted at implementation of security protocols, the design must be based on experience from the implementation and analysis of protocols. The world view taken by BAN can best be seen by examining the notation (symbols) and postulates. The symbols captures *what* in the realm of protocols that is deemed to be of interest (what one can talk about), and the postulates *how* these symbols are related to each other (what one can say). The abstractions offered by BAN has proven themselves to be successful and we take this as our starting point.

Security. In general, determining whether a protocol is secure or not, is undecidable. Even if only finite protocols are considered, and restrictions are placed on the generation of nonces and encryption keys, security is still undecidable [14].

The logics for authentication all make it quite clear that secrecy is a property that can not be analyzed; for a discussion see [15]. From this we conclude that because we are aiming for a versatile programming language, it will be possible to write unsecure protocols in it. In particular, the language will per se not be a tool for the verification or analysis of protocols.

Compatibility. As there's no pretence of a global authority, we can't impose our implementation on all possible participants. Thus our language cannot be based on primitives that has a whole-system view of protocols, and so our approach must deal with protocol endpoints.

We note that there seems to be two different approaches to implementing protocols: implementing existing protocols, and implementing (existing) protocols compatible with existing implementations. The latter seems harder, since it inherently includes compatibility with the implemented protocol's runtime. Because we are concerned with finding the best abstractions to offer to protocol designers, compatibility with existing protocols is not an issue here. In particular: Protocols implemented in general purpose languages (*e.g.* C or Java) can carry solutions we have explicitly left out, preventing interplay, even if the same *functionality* can be supported. We believe that by choosing abstractions with prudence we will have a programming language that can implement all relevant security protocols; this is the same line of arguments as in [13].

Adaptability. Distributed systems are faced with an ever increasing pressure to be adaptable to changes, *e.g.* changes of topology, in the range of equipment, cryptographic technology, and operating systems. The ability to accommodate change will be a necessity; one could say that protocols are too important to be hard-coded into the system. It is not so interesting to discuss what can make it desirable, or necessary, to change a protocols at run time, but a single example should suffice: The decision between public- or shared-key encryption for authentication should not be taken at design time.

Software aging [16] is particularly damaging when it comes to protocols because a change at one place must be reflected throughout the system. And, as we know, protocol design is notoriously difficult and should be avoided if at all possible. If protocols were malleable fewer wide-ranging changes would be necessary.

2.2 Functional Requirements

In deciding what functionality and which abstractions to support in the language we must study the computing model that is to underpin the programming environment.

Cryptography. When discussing the possible design choices, we allow ourself to make the sweeping assumption of perfect cryptography [17,18]. That is: We assume that good engineering can ensure the standard assumptions about encryption holds regardless of which data that is encrypted. Notice that a digital signature in this context is regarded as "encryption" because it produces a result that can not be carried on to other data.

Relevant issues. It must be possible to express *what* to encrypt, *i.e.* indicate which part of a message must be encrypted and which can be sent in clear text. This also applies to signatures and verification, where one also have to deal with computed message parts (*i.e.* message parts not sent).

Which kind of technology also has security implication; *e.g.* whether to use a public or a shared key seems to be crucial. This is not just a parameterization issue, as becomes apparent when asking how a program should deal with the case of moving from public-key to shared-key technology, such as when *e.g.* a PKI becomes unavailable: the protocol must be redesigned to cope with such a change. This in contrast to a change in *e.g.* the cipher mode or padding scheme.

Irrelevant issues. Whether a protocol applies IDEA or AES is irrelevant (from a security point of view). The same goes for issues such as padding and chaining. Such low-level issues are mostly relevant with regard to compatibility between implementations. We do not claim that it is wise to ignore many years of engineering experience and wisdom, so the actual choice of *e.g.* padding or cipher algorithm should be sound, engineering wise, at least. That does *not* mean that the protocol should have to, at a higher level, deal with key initialization, padding sizes or what have you. The protocol should be able to parameterize this to a runtime: define encryption to use some named cipher, and *e.g.* if it is a block cipher, the runtime will provide the IV, insert necessary padding, and so on. The argument that knowing exactly how this is done is paramount for the security actualizations and thus have to be exposed to the programmer only holds if, and only if, the runtime cannot be trusted to both perform the right operations, detect them when conforming, and detect nonconforming behaviour. Exactly what to do, *e.g.* CBC-mode vs. OFB-mode, is a policy decision.

Also, for a system it is essential to determine how a public key is to be verified and validated, but this is definitely a policy issue unrelated to the protocol itself. Hence, there is no support for this in Obol.

Another issue that has become irrelevant is randomness. It seems that all modern processors and operating systems have ample supply of entropy available to programs. Hence, protocols should be expressed and implemented on the assumption that cryptographically nonces and (fresh) keys can be generated at will, and instead we can assume that the runtime will raise an exception (or equivalent) when it can determine that this service is not available.

Naming. All non-trivial protocols assume the existence of a naming scheme, especially in relation to public keys where it must be determined which key to use for a particular purpose.

Naming is inherently a binding to a local identity. By this we mean that a subject is named, and the resulting binding is to some idea of identity that only exists locally where the binding is made. This is even true for SPKI's naming scheme, which allows local names to be used globally [19]. The actual identifying binding only exists when and were it is evaluated, and thus locally.

Exactly what this naming scheme consist of, is unclear in the general case. For instance, a message such as

$$\textit{Message } n \quad A \rightarrow B : A, B, \{A, B, \ldots\}_{K_{ab}}$$

assumes that A, B inside the message, and the encrypted part, somehow identifies the participants A and B. Can A and B be TCP/IP endpoints, or are they named keys in a X509 hierarchy, or MD5-sums of public key certificates, or maybe complete public key certificates? Also, K_{ab} is somehow associated with A and B, so there are actually four ways which the names A and B are used, but it is not at all clear how they are used.

This would indicate that names can have different type, and also different representation. If such representation have security implications for the protocol, then these should be visible when analysing the protocol.

We do not know how to deal with naming in a consistent manner. In particular, it is not clear whether naming is "just" an safety measure or if it has other security implications In fact, we strongly suspect that it is unwise, security wise, not to make the cost associated these naming issues explicitly visible. In the traditional notation names can be sprinkled throughout the protocol [20]. To our knowledge, none of the analysis tools concerns themselves with the representation of names, other than assuming it is possible to generate and parse names, and bind them to an identity: names are treated like symbols, and used only for identification.

One possibility is to use a naming system that facilitates just the binding of local names to global ones; SPKI has been designed with this in mind [21]. However, an implementation of SPKI is itself a non-trivial issue [22].

Naming is one of the issues Obol has been designed to investigate. The crux of the matter is that in the above example, the name "A" is used in four different ways, while it is not at all clear how. We wish to use Obol as our research vehicle to examine this.

Representation of Messages. If the representation of names have no implications for a protocol's security properties, then the representation of any message part have nothing to say for the protocol's security properties. Matters concerning the representation of messages, or message components, are therefore best dealt with by a protocol runtime environment.

Again, this is not an argument for abandoning sound engineering. Our argument that message representation have no consequence for a security protocol's security properties only holds if a protocol runtime is able to "export" security properties, *i.e.* giving guarantees such as ensuring that padding is correctly applied and verified before removal, or that names can be compared. We believe that such a distinction actually enhances the impact of a security protocol, since we now very clearly can specify the requirements under which it will work.

Order of Message Components. An issue which is not at all clear, is whether the order of components in a message has security implications or not. If we for a moment discard the engineering fact that two peers with different assumption on ordering will be unable to communicate, the core of the issue is: Does the order have security implications?

If the answer is affirmative, then, if nothing else, we note that there must be problems applying BAN to the analysis of this protocol. As we know, in BAN the order of components is explicitly not important. On the other hand, if the ordering has no security implications, we can leave it to the runtime to deal with it.

Like naming, ordering of message components is an issue which we will use Obol to examine further.

2.3 Summary

As functional requirements we will demand explicit sending and receiving of messages, encryption primitives, and availability of a high-level naming scheme. There is no need for low-level formatting, padding and other pure engineering aspects. Basically we have used the same line of argument as is used by BAN. The success of BAN demonstrates beyond doubt that important properties of cryptographic protocols are captured by the symbols of the logic, and the rules of interference capture what they mean. The challenge is then to bring the high level abstractions into the real world in a way that is as faithful as possible to both the abstractions, and their realizations.

3 Design of Obol

Having discussed the functional- and non functional requirements of Obol, and our choice of using the world view offered by BAN, we will now discuss the trade-offs when we want to design a language on these terms.

3.1 Layers of Abstraction

Because we want to keep the level of abstraction as high as possible, that is, as close to the language dealing with security properties of a protocol, we cannot concern ourselves with lower-level details not relevant to the security issues.

To make these low-level details reachable, we draw inspiration from the field of middelware, and segregate the different levels of abstraction into components accessible through a well-defined interface. In particular, we want to use the ideas of reflection and composition to achieve a runtime supporting pluggable low-level functionality. This means that we treat message representation and transformations up and down various layers of abstraction as pluggable components. See Figure 1.

The goal is to use the same model of interaction between high-level language constructs and low-level functions, between protocols at the high level of abstractions, so that protocols can use other protocols, as well as using this model for accessing the runtime functionality from an application's point of view.

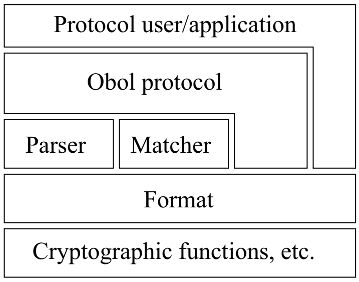


Fig. 1. Architecture. The Format layer deals with message representation, while the Parser and Matcher layers deals with message composition and recognition. In implementations, some overlap of these three layers will occur.

3.2 Syntax and Interpretation

We want the syntax of a protocol language to simple enough so that writing a parser should be straight forward, with as few terminals in the language as possible. To that end, we borrowed ideas from Lisp, so we have a functional prefix Lisp-like syntax, where program and data share the same structure, and where symbols are placeholders for properties and values, which have type. We investigate some of the language's primitives below.

A program written in Obol consists of a list of statements. Statements returns values that are implicitly assigned to the context in which the statement is executed. The language is not purely functional, however, as there are calls that are called solely for their side effects.

Obol is not a logic, and beliefs need not remain true as they do in a modal logic. In particular, beliefs need not be stable [15].

3.3 Language Constructs

An implementation of a protocol has to deal with a given participant's point of view, *i.e.* it only describes one side of the protocol. This means that for each participant in a protocol there exist a local view of the protocol state, and every statement regarding other participant's view of the protocol state will be guesswork. The design of the protocol as a whole serves to make this guess as good as possible. However, an implementation of a particular participant's rôle in the protocol cannot allow itself to speculate on the validity of remote state, and can only consider it's own state with certainty. This local state can only be updated by a participant's explicit actions, that is, what it contributes by itself, or what it accepts from "outside." In reality, only the receipt of a message warrants a state change as sending a message does not imply it will be received. The primitives of a protocol language must reflect this.

There are constructs not discussed here, including operands for loading and saving data to files, returning protocol results, attaching protocols to representation-level components, looping, error and exception handling. We believe none of them have security implications (outside the realm of engineering).

Local State. There are two primitive operators that changes the local state of a protocol implementation end-point: `believe` and `generate`.

Our world view is that which is entrenched in BAN. The principal *believes* something about his own state, and that of his peers. The `believe` operator updates local state updated with some given information about some symbol. We allow symbols to have multiple values of multiple types, which allows us to deal with complex symbols such as names, where *e.g.* A can designate both a TCP-connection and a textual name.

The `generate` operator updates the local state by constructs new data of a specified type. Because the runtime is trusted to provide data of high enough quality, generated data is also implicitly believed to be good:

```
(generate R shared-key AES 128)
```

Here a 128-bit AES-key is generated and assigned to a variable named R.

The programmer can change what is believed about a given symbol:

```
(generate Q nonce 128)
(believe K Q ((type shared-key)(alg AES)))
```

Here a 128-bit nonce is generated and assigned to symbol Q. The content of Q is then explicitly believed to be good as an encryption key and then stored in the variable K. The reason for storing the information with two types might be that it needed both as a nonce and as a key.[1]

Having access to a key does not imply that any particular beliefs are held about the key, except that the bits indeed do represent a key. In BAN, no distinction is made between having a key and having belief in it; this is altered in later efforts along the same lines [23]. Because Obol is a programming language, it is the programmer that decides the beliefs he deems it justified to hold on a key. The language only enforces that beliefs on the type must be made explicit.

Sending and Receiving Messages. The `send` primitive constructs a message from a list of symbols (or the primitives `encrypt` and `sign`) and sends it to a known protocol participant. It is left to the runtime to implement an encoding of the components that makes parsing and reconstruction possible. No assumptions are made on the properties of the communication primitives used to implement message transmission. If, for example, ordering of messages is important in the protocol (which is almost always the case) elements must be added (by the runtime) to the messages to ensure correct ordering.

The only way of being influenced by other participants in a protocol, is to receive messages from them. The `receive` primitive takes a specification (a *message template*) of what the expected message is supposed to contain, using symbols and primitive operations that are the converse of those used by the `send` primitive. In addition to recognizing known message components, we "recognize" previously unknown ones by assigning them to anonymous variables (symbols

[1] Which is suspicious from a security point of view, but here we are just demonstrating the power of the language.

prefixed with an asterisk, *e.g.* "`*A`"). The anonymous variable can later be given type using the `believe` primitive, or it could just be kept as an opaque binary data object.

In the following protocol, a nonce is generated and sent. A reply is then expected containing the same nonce, and a peer-provided new one.

```
(believe A "somehost:1234" host)
(generate Na nonce 128)
(send A Na)
(receive A Na *Nb)
```

The call to `receive` blocks, and returns when a message containing both the nonce and a unknown component is received. The message template is here only a nonce and an anonymous variable; more complex patterns are possible when dealing with encrypted material (see below). The programmer gives the system a hint on where to look for the message, instead of having the runtime inspecting all incoming communication, which can have severe performance implications if the inspection involves decryption or verification. If the programmer don't know or care where a message would come from, an anonymous variable could be used instead, assuming the runtime supports arbitrary reception of messages (which can be the case in *e.g.* a middelware infrastructure).

Following our design of considering only one side of the communication, receiving a message is a local event. As such, the receipt of a message is not detectable by others. This view of message passing is in accordance with the model of computation found in [23, Sec. 5].

Cryptographic Operators. There are two classes of cryptographic operators, for message transformation, and for verification. Message transformation is done with the `encrypt` and `decrypt` operators. They are similar to `send` and `receive`, but the specified message is sent to/received from a key.

The `encrypt` operator takes a key and a message specification, and yields a binary ciphertext message component. Encryption is parameterized by the key, so believing something about the key changes the result of using it (*e.g.* changing the associated cipher algorithm).

Like `receive`, the `decrypt` operator accepts a message template for what is expected to be "received" by decryption, as well as assignment of unknown message components. The input ciphertext is either provided by the environment (line 1), or explicitly stated by the programmer (line 2):

```
1. (receive A (decrypt k "foo" *F4))
2. (decrypt (k (encrypt k "foo" 65537)) *1 65537)
```

Message verification is supported by the operands `sign` and `verify`, which accepts a key and a message specification, and either produces or checks a signature value.

```
(verify (Kpublic (sign private "foo" 65537)) "foo" 65537)
```

Like `receive`, `verify` can obtain the signature value to check either explicitly or implicitly from the context. Verification failure outside the context of `receive` is an error condition, while inside it means that the message wasn't received.

In all signature schemes we are aware of, one must explicitly specify what the signature is on. Also, the signature can be on data that is not actually sent, but computed. We believe that the `sign` and `verify` operators captures this explicitness. Even if the trade-off is repetition of message specifications, we cannot see a better (*i.e.* more elegant and general) way of doing this.

4 Implementation

4.1 Prototypes

We have several implementations of our ideas, although none are complete.

We started using Common Lisp to do a rapid prototype – this has also obviously influenced Obol syntax, and we got inspired by the dynamic programming environment.

Since one of the main application domains seems to be middelware (see Section 5.1), effort was invested into applying Obol in one of the more experimental reflective middelware platform: OOPP[24].

To test our hypothesis that the Lisp-inspired roots of Obol didn't influence our ideas, an imperative-style version was implemented.

The latest version of Obol returned to the original "Lisp-ish" notation, but implemented the runtime in Java, the goal being integration in an industrial middelware platform (JBOSS).

We use the name "Lobo" for a given implementation of the Obol runtime.

4.2 Prototype in Java

Implementing an Obol runtime in Java allows us to demonstrate applicability, as well as explore how to best utilize Obol/Lobo from a protocol user (*i.e.* application) standpoint. Our current approach is to instrument Lobo using the Java Management Extension (JMX), and is ongoing work. A schematic over the current design structure can be found in Figure 2.

The default message representation is the Java Serialization format. An example test-run from the Java environment using the Serialization format is shown in Figure 3. Although Serialization allows for all kinds of interesting behaviour, such as sending classes and other complex objects, it would appear that most security protocol messages consists of primitive data types, *e.g.* integers, strings and byte-arrays. Since the default transport is Serialization, we believe it will serve as a useful measure for how difficult it will be to provide other representation formats. If a protocol requires a format as powerful as Java Serialization, then it would endanger our hypothesis that a protocol's representation format is irrelevant. That said, we do see possibility of constructing a protocol that only works using Java Serialization, but we argue that in such a case one is actually

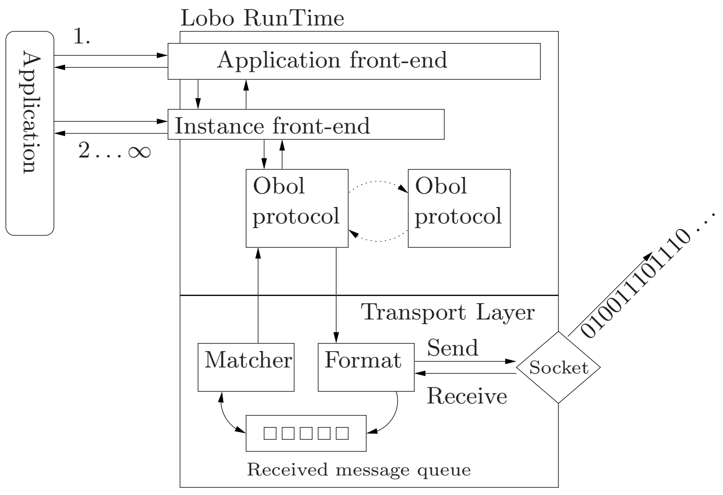


Fig. 2. Java Lobo implementation structure. The application first contacts the application front-end to obtain a binding to a protocol instance, and interacts with this instance from then on.

```
% (generate k shared-key AES 128)
## "k" = "javax.crypto.spec.SecretKeySpec@17720",
    (("generation spec" "shared-key AES 128")
     ("defined by" "generate")("defined line-number" "1"))
% (believe *a (encrypt k "foo" "bar" 65537))
## "*a" =
     0: 165C 5484 41B1 0F3F AB9D E834 3D63 1E1F   ..T.A.....4.c...
    16: A1E6 4267 D5F8 2831 D03B 6D1F 5739 45B2   .¾BgÍ¿.1?.m.W9E.
    32: 45C9 9C36 AFF8 08F3 E1CF 3DF2 9194 D306   E.6.¿.ì.Ëê......
    64: 19BB 05AA 289D 0065 2EF0 F70B 8902 B087   .......e.?......,
    (("type" "binary")("defined by" "believe")
     ("defined line-number" "2"))
% (decrypt (k *a) *1 "bar" *3)
% (believe *3 ((type number)))
## "*3" = "65537",
    (("changed at line" "4")("type" "number")
     ("changed by" "believe")("defined by" "decrypt")
     ("defined line-number" "3")("number-of-bits" "32"))
```

Fig. 3. Verbose debugging output from a test-run of Java implementation of the Obol runtime

focusing on achieving compatibility with an existing application, which is not the focus of our efforts.

The Java prototype is currently the most active Obol project.

4.3 Examples

Below are some examples of protocols implemented in Obol; they all run on the current implementation of Lobo.

Implementation of A:

```
 1 [self "host-A:9000"]
 2 (believe B "host-B:9000" host)
 3 (believe Kas (load A)
      shared-key ((alg AES)))
 4 (generate Na nonce 128)
   ;; Message 1
 5 (send B A Na)
   ;; Message 3
 6 (receive *S (decrypt Kas
       B *Kab Na *Nb) *toB)
 7 (believe Kab *Kab
      ((type shared-key)(alg AES)))
   ;; Message 4
 8 (send B *toB (encrypt Kab *Nb))
```

Implementation of B:

```
 1 [self "host-B:9000"]
 2 (believe S "host-S:9000" host)
 3 (believe Kbs (load B)
      shared-key ((alg AES)))
   ;; Message 1
 4 (receive *A *Na)
 5 (generate Nb nonce 128)
   ;; Message 2
 6 (send S B (encrypt Kbs *A *Na Nb))
   ;; Message 4
 7 (receive (decrypt Kbs *A *Kab) *2)
 8 (believe Kab *Kab
      ((type shared-key)(alg AES)))
 9 (decrypt (Kab *2) Nb)
```

Implementation of S:

```
 1 [self "host-S:9000" host]
   ;; Message 2
 2 (receive *B *fromB)
 3 (believe Kbs (load *B) shared-key ((alg AES)))
 4 (decrypt (Kbs *fromB) *A *Na *Nb)
 5 (believe *A ((type host)))
 6 (believe *Na ((type nonce)))
 7 (believe *Nb ((type nonce)))
 8 (believe Kas (load *A) shared-key ((alg AES)))
 9 (generate Kab shared-key AES 128)
   ;; Message 3
10 (send *A (encrypt Kas  *B Kab *Na *Nb) (encrypt Kbs  *A Kab))
```

Fig. 4. An Obol implementation of the Yahalom protocol

Yahalom. The Yahalom protocol (taken from [25, page 30]) has four messages as follows:

$$\begin{array}{ll} \textit{Message 1} & A \rightarrow B : A, N_a \\ \textit{Message 2} & B \rightarrow S : B, \{A, N_a, N_b\}_{K_{bs}} \\ \textit{Message 3} & S \rightarrow A : \{B, K_{ab}, N_a, N_b\}_{K_{as}}, \{A, K_{ab}\}_{K_{bs}} \\ \textit{Message 4} & A \rightarrow B : \{A, K_{ab}\}_{K_{bs}}, \{N_b\}_{K_{ab}} \end{array}$$

Figure 4 shows an Obol implementation of this protocol; the line numbers are not part of the implementation, and the numbers in comments (non-numbered lines starting with “;;”) refer to the protocol description above.

Notice in the implementation of S that we have chosen to promote the two components into nonces (lines 6 and 7). This is not strictly necessary, as they could have been included in the encryption expression in the line 10 without having been promoted. However, by promoting them we give the runtime the possibility to examine them as nonces (and not just data). This can be used to verify that they haven’t been seen before, that they seem to be random, and so on. This illustrates how one with ease can experiment with protocols and their implementation when the protocol has been implemented in Obol.

Implementation of A:

```
[self "host-A:9000"]
(believe B "host-B:9000" host)
(believe P (load "P.key")
    shared-key ((alg AES)))
(generate Kt keypair RSA 512)
(send B A (encrypt P
  (public-key Kt)))
(receive (decrypt P
  (decrypt (private-key Kt) *Kab)))
(believe Kab *Kab
  ((type shared-key)(alg AES)))
```

Implementation of B:

```
[self "host-B:9000"]
(believe P (load "P.key")
    shared-key ((alg AES)))
(receive *A (decrypt P *Kt))
(believe *Kt ((type public-key)
              ((alg RSA))))
(generate Kab shared-key AES 128)
(send *A (encrypt P
           (encrypt Kt Kab)))
```

Fig. 5. An Obol implementation of the two first messages in the EKE protocol [26]

EKE. EKE (Encrypted Key Exchange) was proposed by Bellovin and Merritt [26] to secure password-based protocols against dictionary attacks. It also achieves perfect forward secrecy, that is, the disclosure of the password (long term secret) does not compromise the session key produced by the protocol.

Let A and B be the two principals in the protocol, P a shared long term secret (password or secret key), and K_t the public part of a temporary public key pair. The protocol consists of five messages where the last three are for mutual verification of the key; we present only the two first which show the essence of the protocol:

$$\begin{array}{ll} \textit{Message 1} & A \rightarrow B : A, \{K_t\}_P \\ \textit{Message 2} & B \rightarrow A : \{\{K_{ab}\}_{K_t}\}_P \end{array}$$

The Obol implementation is shown in Figure 5.

4.4 Conclusion

The implementation of Obol is fully functional, is written in 100% pure Java, and can be embedded in applications and systems. The latest version can be obtained by contacting the authors. The small examples shown here gives a mere flavor of the language. We believe that almost all of the protocols in the Clark-Jacob library[2] can be implemented in Obol.

5 Application

As Obol makes it possible to write and deploy protocols it is prudent to ask where it is reasonable to apply them. Because Obol cleanly decouples protocol implementation from protocol execution we can apply heavy-weight protocols in settings where they normally are outside reach. We have identified three challenging settings:

[2] An updated version of this collection is currently available at http://www.lsv.ens-cachan.fr/spore/

1. Providing configurable protocols to components in a middleware system;
2. Indirectly executing heavy weight protocols on smartcards;
3. Revocable protocols; infrastructure to ensure that a service provider can revoke a protocol.

Below we will elaborate on these applications of Obol.

5.1 Middleware

General middleware environments offer an execution environment for components, usually called a capsule or a container. Components implement the business logic of applications, while the container provides logging, communication facilities, and so on. This computational model works well when the (set of) components together with the container, are self-contained. The situation becomes murky when components need to rely on external services, in particular when these services must be accessed in a secure manner. Such services might be to convey funds, make reservations, and so on. The problem is the inherent complexity in security protocols, be they advanced authentication technology or simple encryption, where keys must be obtained from key distribution (*e.g.* certificate) infrastructures. Regardless of purpose or origin, code necessary to conform to a particular cryptographic regime must be implemented on the client side. The problem is then how to easily adapt components to changes in the interfaces offered by services, and still avoid that all components must carry the code to execute all these protocols. For example, if a server changes from SSH-1 to SSH-2 or from SSL to TLS, this should be a minor change; it does not at all change the business logic. However, if a new implementation must be provided at the client side, it is a major obstacle. There are two possibilities: Either change the deployment descriptor of the component and redeploy, or rely on the (provider of the) container to provide the code. Since both approaches have obvious disadvantages, the net effect is a striking lack of flexibility, very much in contrast to the original goals that made this programming model attractive in the first place.

By using Obol, components can implement cryptographic protocols without having to rely on particular protocol implementations in the container. Also, because Lobo would become the focal point of a variety of protocols, analysis of their implementation becomes feasible.

Components confined to a container communicate with external services by means of Lobo. Lobo might itself be implemented as a component, and provide an interface where other components (called clients) download programs (protocols) to be executed by the coprocessor. Other implementations are also possible, *e.g.* LOBO could be part of the container implementation, or be offered as an external service accessible through the container.

One of the main problems is that we need a secure channel to LOBO. A component running in a container will have to trust the container to provide this channel, in effect incorporating the container into the component's trusted computing base (TCB [5]), a scenario which is true in any case. If LOBO is located outside the immediate environment, a secure channel has to be established first using some technique for authentication and privacy.

5.2 Smartcards and PDA

Another setting where Obol can be applied is systems that use equipment with limited processing, storage, and communication facilities, such as smartcards. Here the problem stem from the challenge of reprogramming a large number of cards, but also from the fact that the resources available in smartcards are scarce. Obviously this holds for all kinds of devices deprived of resources.

In the middleware setting, communication between LOBO and the component is provided by the container. In a setting with smartcards, a shared key can be installed in the card and used to establish a secure channel to a server (which is part of the TCB). The shared key, when trusted by both sides, represents a trusted channel that provides both authentication and privacy. The program, written in Obol, is then transferred from the card to the server, the server executes the program, and sends the results back over the channel realized by the shared key. Knowing how scarce resources are in smartcards, using LOBO and Obol might make new and exciting applications possible [27].

Obviously, applying Obol and Lobo to leverage a server does not solve the fundamental problem that a weak machine isn't able to perform all the tasks the user would like. In particular, the TCB increases which is a problem not to be viewed favorably. On the other hand, delegating authority (over protocol execution) is an extension of the TCB that is not so hard to analyze.

The great advantage of smartcard is their tamper-resistance, and that this makes it possible to build a TCB that is distributed. In most cases, physical access to the computing device makes it impossible to trust its integrity, but this is normally not the case with smartcards. This, however, is countered by their lack of flexibility. They are hard to program, and the trust model surrounding their use is complex. In particular, smartcards without a channel on which feedback can be offered to users are prone to a wide range of attacks that are hard to defend against [28,29].

By using Obol, applications on the card can engage in more complex protocols without having to have the full implementation on board.

5.3 Certificates

A third setting is one where a service can use existing infrastructure for authentication, that is, use *e.g.* an X.509 or SPKI-based PKI to distribute certificates. The server can embed in the certificates an Obol program that, when executed, realizes the client side of the protocol on which the service is provided. The client would then obtain the certificates, verify the correctness, and immediately have access to an authenticated version of the protocol needed to access the service.

In addition to allowing us to efficiently distribute a protocol, this approach also makes it possible to revoke protocols when security considerations makes this desirable.

Having revocable protocols as part of certificates is novel, and demonstrates the effectiveness of Obol as a research vehicle.

5.4 Conclusions

The overall goal of Obol is to facilitate an experimental platform for protocol research. The three, very different, areas of applications demonstrates fully that Obol can be applied in a variety of settings.

6 Related Work

The effort of Obol draws on projects from four different categories: logics, formal description techniques, protocol implementation languages, and middleware solutions.

6.1 Logic

Various methods can be used to analyze protocols to determine whether they achieve their goals, in the hope that the designers can convince themselves that the protocol's assumptions, state transitions and desirable goals are sound and attainable [30]. These analysis tools have also successfully been used to find and prove design flaws in existing protocols, often in protocols that were believed to be sound [31]. There are several classes of tools available for such analyses, ranging from modal logics [12,32,33], to Higher Order Logics rewrite systems [34], and to state attainability deducers [30,31]. Common to all these approaches is that they prove or disprove the reachability of a protocol's goal state from its initial assumptions via a set of transitions, and that they operate on a description of protocols that is not executable.

Obol is definitely not a logic, but rather a high-level programming language. However, the abstractions the language offer has been chosen to make the transition from an analyzed protocol to implementation as simple as possible. The relations to BAN are obvious, and the *raison d'êntre* for Obol is, after all, to give those working at a higher level of abstraction the means to implement their protocols and being reasonable confident that the implementation adheres to the protocol description used in the analysis.

6.2 Formal Description Techniques

LOTOS (Language Of Temporal Ordering Specification) was developed by OSI in the late eighties as a Formal Description Technique [35,36]. It is a language for the description of protocols. It has complementary formalisms for 'data' based on ACT ONE [37] and 'control' based on CSP [38]. Estrel is one of the family of languages used to describe real-time systems using the state machine model [39,40].

This approach is fundamentally different from the one taken in Obol. A protocol implemented in Obol is not a vehicle for analysis but rather a language as close to the original specification as possible.

Another approach is to assume that TCP/IP is available, and then place a new layer between the application and the transport layer [41]. This new layer is then responsible for negotiating security properties between the parties. The system, named LEI (Logical Element of Implementation) can interpret and implement any security protocol from its specification. As is custom for these approaches, there is only one protocol description and both sides need to run identical software. In some sense this approach could be classified as middleware.

Obol is related also to CAPSL, which is a high-level language for applying formal methods to the security analysis of cryptographic protocols. Its goal is to permit a protocol to be specified once in a form that is usable as an interface to any type of analysis tool or technique, given appropriate translation software [42]. Protocols specified in CAPSL are translated into an intermediate format (named CIL), and from there to any format needed by the tool that is to be used. Related to Obol, there are tools that from CIL generates Java [43] and Athena [44,45]. Capsl features a clearly defined notion of types, encryption and the sending and reception of messages. This project has extended the effort also to analyse secure multicast protocols [46].

Although related, Obol differs substantially from CAPSL in that the latter is designed to assist in the analysis of protocols, rather than be executed "as is."

6.3 Implementation Languages

Prolac is a language for implementation of protocols [47]. The language can be compiled into C, and, as is the case with Obol, both sides need not be implemented in the language. Prolac compiles to C and the idea is that the resulting code can be embedded in an application or system. Because the language has been designed to facilitate the implementation of "TCP-like" protocols, Prolac has support for low-level issues such as buffer management, and the inclusion of code written in C. Both of these would thwart the efforts of Obol to ensure that such low-level issues do not pollute the protocols.

A very different approach is to start out with cryptographic primitives and generate a protocol from the elements used in the encryption [48]. The resulting protocol, which is to be embedded in both sides of the communication, is claimed to be provable secure because it is derived from a mathematical problem witch is provable hard. Such an approach is very limited compared to Obol, and there will be no flexibility on how the protocol is to act.

The design of Obol has been guided by our desire to capture in a programming language powerful abstractions from the domain of security protocols. The language Morpheus was designed in precisely the same manner, but for a different setting [13]. Also this project aims for the implementation of "TCP-like" protocols, and as such only the methodology of the project is directly relevant for us. An interesting observation regarding Morpheus is that the authors claims that Morpheus is situated between the two extremes represented by Formal Description Techniques (see above) on one end, and implementations in a general-purpose language constrained only by the operating system on the other [13, Section II] This is exactly the same position as we claim for Obol, but in the security protocol domain. Morpheus was not fully implemented.

6.4 Middleware

Middleware solutions that are related to Obol can be divided in two: Systems that composes protocols to achieve the sough-for properties, and systems that alter encryption properties to achieve the same.

Protocol Composition Systems. One can reasonable view Obol as middleware: A software layer that offers services to applications. In the case of Obol the service is mechanisms to construct (security) protocols. Because Obol is concerned with the construction and execution of protocols it is related to systems where an application can use underlying mechanisms to construct protocols, for example Ensemble [49] (there are others *e.g.* Antigone [50]).

However, the focus is on security *properties* that can be achieved and maintained within group communication by applying protocols, rather than how to construct these protocols. The target of Obol can be viewed as more focused, in that we are concerned with the implementation of protocols rather than their composition.

Interceptors. Both in .NET and in EJB anyone (with sufficient privilege) can alter the protocol by inserting in the stream of data so called *interceptors*. In some senses this is protocol programming. The mechanisms are designed to alter (for example by encrypting) a stream of data passing from one peer to another, and not to change the protocol all together.

"Da Capo++" is a middleware system where many of the application's needs and communication demands can be specified in terms of QoS values [51]. Da CaPo++ has a well defined protocol machinery (named Lift) which is at the core of the system. The data is managed by the Lift, and passed to modules that are inserted (or removed) according to the QoS specifications of the application.

Da CaPo++ also places security under the same QoS regime as other resources available to the system. The means that the "degree of privacy" has to be specified as a QoS value (a number). The problem is, obviously, that any number needs a semantic mapping to be useful.

Da CaPo++ and Obol are very different systems. Da CaPo++ is *one* system that is intended to run on both sides of a communication channel; it is middleware. Obol, on the other hand, is a subsystem for protocol execution, and a protocol realized with a program written in Obol can communicate just as well with a system that does not run Obol.

Even though the systems are very different, there are two aspects where Da CaPo++ is relevant for Obol. First, Da CaPo++ views security as one of many QoS attributes, and there are mechanisms to manipulate security attributes in the same manner as one manipulates other relevant aspects of the system. If Obol was integrated into Da CaPo++ the resulting system would be even more flexible than what Da CaPo is today. Rather than mainly altering encryption algorithms, Da CaPo++ could also freely change the protocol. Second, in general Obol programs do not carry instructions for the run time on which algorithms to choose, but this could conveniently be done by the mechanisms offered by Da CaPo++.

6.5 Jini

Jini is a network technology supplied by Sun as part of the Java effort. It provides infrastructure components and programming models for allowing services to be discovered by clients, and providing these clients with the necessary executable code in order to use the services.

Albeit superficially similar to Obol, in that code is made available to clients, the important distinction is that Jini is essentially a service-“driver” distribution facility, heavily relying on a Java infrastructure for code (classes) distribution, reflection and usage. It does not concern itself with protocols per se, other than possibly providing a proxy class for accessing a service over a particular protocol. It can also be argued that Jini provides implementations for both sides of the protocols it provides.

7 Discussion

The abstractions offered by Obol has been chosen so that the functionality of a majority of security protocols can be implemented. At the same time, Obol avoids making any global assumptions, and all security related issues are left to the programmer; we believe this is sound engineering. Part of this is to ensure that the protocol programmer does not control the ordering of components of messages, and thus does not rely a particular ordering for security.

Obol exposes in full the intricate issue of naming; an Obol program has embedded in it the naming scheme of the system, which we believe is sound engineering. So called SPKI names is but one possibility.

The even increasing diversity in computing milieus has led us to design Obol as a language in it’s own right rather than as a set of libraries (classes) for a particular language, middleware platform or operating system.

As Obol is underpinned by an implementation this makes it possible experiment with protocol design. Furthermore, as all packages from all instances of protocols passes through the same machinery, monitoring for intrusion detection and replay attacks can be put in place.

References

1. Anderson, R., Needham, R.: Programming satan’s computer. In: van Leeuwen, J. (ed.) Computer Science Today. LNCS, vol. 1000, pp. 426–440. Springer, Heidelberg (1996)
2. Barak, B., Halevi, S., Herzberg, A., Naor, D.: Clock synchronization with faults and recoveries (extended abstract). In: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing, pp. 133–142. ACM Press, New York (2000)
3. Zhou, L., Schneider, F.B., Renesse, R. V.: Coca: A secure distributed online certification authority. ACM Trans. Comput. Syst. 20(4), 329–368 (2002)
4. Department of Defense: DoD 5200.28-STD: Department of defense (DoD) Trusted Computer System Evaluation Criteria (TCSEC), The Orange Book (1985)

5. Lampson, B., Abadi, M., Burrows, M., Wobber, E.: Authentication in distribued systems: theory and practice. ACM Transactions on Computer Systems 10(4), 265–310 (1992)
6. Thompson, K.: Reflections on trusting trust. Communications of the ACM 27(8), 761–763 (1984); Also appears in ACM Turing Award Lectures: The First Twenty Years 1965-1985. ACM press, New York (1987), and Computers Under Attack: Intruders, Worms, and Viruses. ACM press, New York (1990)
7. Simmons, G.J.: Cryptanalysis and protocol failures. Communications of the ACM 37(11), 56–65 (1994)
8. Stubblefield, A., Ioannidis, J., Rubin, A.D.: A key recovery attack on the 802.11b wired equivalent privacy protocol (wep). ACM Transactions of Information Systems Security 7(2), 319–332 (2004)
9. Abadi, M., Needham, R.: Prudent engineering practice for cryptographic protocols. IEEE Transactions on Software Engineering 22(1), 6–15 (1996); A preliminery version appeared in the Proceedings of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy (1994)
10. Harris, J., Henderson, D.: A better mythology for system design. In: ACM Conference on Human Factors in Computing Systems, pp. 88–95 (1999)
11. Anderson, R., Needham, R.: Robustness principles for public key protocols. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 236–247. Springer, Heidelberg (1995)
12. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. ACM Transactions on Computer Systems 8(1), 18–36 (1990)
13. Abbott, M.B., Peterson, L.L.: A language-based approach to protocol implementation. IEEE/ACM Transactions on Networking 1(1), 4–19 (1993)
14. Durgin, N., Lincoln, P., Mitchell, J., Scedro, A.: Undevidability of bounded security protocols. In: Heintze, N., Clark, E. (eds.) Proceedings of the Workshop on Formal Methods and Security Protocols, Trento, Italy (1999)
15. Syverson, P.F.: Knowledge, belief, and semantics in the analysis of cryptographic protocols. Journal of Computer Security 1(3), 317–334 (1992)
16. Parnas, D.: Software aging. In: Proceedings of the 16th international conference on Software engineering, Sorrento, Italy, pp. 279–287 (1994)
17. Blum, J.R., Goldwasser, S.: An efficient probabilistic public-key encryption scheme which hides all partial information. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 289–302. Springer, Heidelberg (1985)
18. Goldwasser, S., Micali, S.: Probabilistic encryption and hos to play mental poker. In: Proceedings of the 14th ACM Symposium on the Theory of Computing (1982)
19. Halpern, J.Y., van der Meyden, R.: A logical reconstruction of SPKI. Journal of Computer Security 11(4), 581–613 (2004)
20. Abadi, M., Needham, R.: Prudent engineering practice for cryptographic protocols. IEEE Transactions on Software Engineering 22(1), 6–15 (1996)
21. Halpern, J.Y., van der Meyden, R.: A logic for SDSI's linked local name spaces. In: PCSFW: Proceedings of The 12th Computer Security Foundations Workshop. IEEE Computer Society Press, Los Alamitos (1999)
22. Myrvang, P.H.: An infrastructure for authentication, authorization and delegation. Cand. scient. thesis, Dept. Computer Science, University of Tromsø, Norway (2000)
23. Abadi, M., Tuttle, M.: A Semantics for a Logic of Authentication. In: Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing, pp. 201–216 (1991)

24. Andersen, A., Blair, G.S., Eliassen, F.: A reflective component-based middleware with quality of service management. In: PROMS 2000, Protocols for Multimedia Systems, Cracow, Poland (2000)
25. Burrows, M., Abadi, M., Needham, R.: A logic of authentication, from proceedings of the royal society. In: Stallings, W. (ed.) Practical Cryptography for Data Internetworks, vol. 426(1871). IEEE Computer Society Press, Los Alamitos (1989)
26. Bellovin, S., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland (1992)
27. Rankl, W., Effing, W.: Smart Card Handbook, 2nd edn. John Wiley & Sons, Chichester (2000), ISBN 0-471-98875-8
28. Abadi, M., Burrows, M., Kaufman, C., Lampson, B.: Authentication and delegation with smart-cards. Science of Computer Programming 21(2), 93–113 (1993)
29. Stabell-Kulø, T., Arild, R., Myrvang, P.H.: Providing authentication to messages signed with a smart card in hostile environments. In: Proceedings from the USENIX Workshop on Smartcard Technology, pp. 93–99 (1999)
30. Meadows, C.: Formal Verification of Cryptographic Protocols: A Survey. In: Safavi-Naini, R., Pieprzyk, J.P. (eds.) ASIACRYPT 1994. LNCS, vol. 917, pp. 133–150. Springer, Heidelberg (1995)
31. Meadows, C.: The NRL Protocol Analyzer: An Overview. The Journal of Logic Programming 26(2), 113–131 (1996)
32. Gong, L., Needham, R., Yahalom, R.: Reasoning about Belief in Cryptographic Protocols. In: Proceedings of the IEEE 1990 Symposium on Security and Privacy, Oakland, California, pp. 234–248 (1990)
33. Syverson, P.F., van Oorschot, P.C.: A unified cryptographic protocol logic. CHACS Report 5540-227, Naval Research Laboratory, Washington, USA (1996); Parts of this paper appeared in preliminary form in [52] and [53]
34. Brickin, S.H.: Automatically detecting most vulnerabilities in cryptographic protocols. In: DARPA Information Survivability Conference and Exposition, Hilton Head Island, SC, USA (2000)
35. Bolognesi, T., Brinksma, E.: Introduction to the ISO specification language LOTOS. In: van Eijk, P.H.J., Visser, C.A., Diaz, M. (eds.) The formal description technique LOTOS, pp. 23–73. North-Holland, Amsterdam (1989)
36. ISO: Information processing systems — Open systems interconnection — Estelle — a formal description technique based on an extended state transition model (1989)
37. Ehrig, H., Mahr, B.: Fundamentals of Algebraic Specification 1: Equations and Initial Semantics. Springer, Heidelberg (1985)
38. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs (1985)
39. Boussinot, F., de Simone, R.: The ESTEREL language. IEEE Transactions on Software Engineering 9(79), 1293–1304 (1991)
40. Berry, G., Gonthier, G.: The ESTREL synchronous programming language: Design, semantics, implementation. Science of Computer Programming 2(19) (1992)
41. Mengual, L., Barcia, N., Jiménez, E., Menasalvas, E., Setién, J., Yágüez, J.: Automatic implementation system of security protocols based on formal description techniques. In: Corradi, A., Daneshmand, M. (eds.) Proceedings of the Seventh IEEE Symposium on Computers and Communications, pp. 355–360. IEEE Computer Society, Los Alamitos (2002)

42. Brackin, S., Meadows, C., Millen, J.: Capsl interface for the nrl protocol analyzer. In: Proceedings of the Symposium on Application - Specific Systems and Software Engineering and Technology, pp. 64–73. IEEE, Los Alamitos (1999)
43. Millen, J., Muller, F.: Cryptograpic protocol generation from capsl. SRI Techical Report SRI-CSL-07-01, Computer Science Laboratory, SRI international (2001)
44. Perrig, A., Phan, D., Song, D.X.: ACG-automatic code generation. Automatic implementation of a security protocol. Techical Report 00-1120, UC Berkeley (2000); This technical report was never issued
45. Perrig, A., Song, D.: A first step towards the automatic generation of security protocols. In: Network and Distributed System Security Symposium, NDSS 2000, pp. 73–84 (2000)
46. Millen, J., Denker, G.: Mucapsl. In: DISCEX III, DARPA Information Survivability Conference and Exposition, pp. 238–249. IEEE Computer Society, Los Alamitos (2003)
47. Kohler, E., Kaashoek, M.F., Montgomery, D.R.: A readable TCP in the Prolac protocol language. In: ACM SIGCOMM, pp. 3–13 (1999)
48. MacKenzie, P., Oprea, A., Reiter, M.K.: Automatic generation of two-party computations. In: Proceedings of the 10th ACM conference on Computer and communication security, Washington D.C., USA, pp. 210–219 (2003)
49. van Renesse, R., Birman, K.P., Maffeis, S.: Horus: A flexible group communication system. Communications of the ACM 39(4), 76–83 (1996)
50. McDaniel, P.D., Prakash, A., Honeyman, P.: Antigone: A flexible framework for secure group communication. In: Proceedings of the 8th USENIX Security Symposium, pp. 99–114 (1999)
51. Stiller, B., Class, C., Waldvogel, M., Caronni, G., Bauer, D., Plattner, B.: A flexible middleware for multimedia communication: Design implementation, and experience. IEEE JSAC: Special Issue on Middleware 17(9), 1614–1631 (1999)
52. van Oorschot, P.C.: Extending cryptographic logics of beliefs to key agreement protocols (extended abstract). In: Proceedings of the First ACM Conference on Computer and Communication Security, pp. 232–243 (1993)
53. Syverson, P.F., van Oorschot, P.C.: On unifying some cryptographic protocol logics. In: Proceedings of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy, Los Alamitos, California, USA, pp. 14–28. IEEE Computer Society Press, Los Alamitos (1994)

Cordial Security Protocol Programming
(Transcript of Discussion)

Tage Stabell-Kulø

University of Tromsø

Chris Mitchell: Arguably, different encryption primitives have different properties, for example they may or may not offer non-malleability. That's an important distinction, because some protocols require non-malleable encryption, and some don't.

Reply: Yes, this is a good point. I have been told by my advisor earlier in my life, never say, this is a good question, because you must assume that, but anyway, this is a good question. If you want everything, you end up getting nothing, and you will see we have found the trade-off, and I will return to the metric by which we measured the trade-off, but there will always be cases which cannot be met, because this is a programming language.

George Danezis: Isn't there a problem for an application protocol, because basically you're saying that we already rely on the authentication for structure, but none of the curly-brackets-notation protocols travelling back and forth are actually to establish the identity of the user.

Reply: The fact that Alice and Bob cannot escape their names, makes the public key system work, so naming, in itself, has very important security implications. Therefore, a language for protocol design must make explicit the naming system that you use, and if your protocol assumes that some globally consistent namespace is available, well then that must be visible in your code. When we started on this, we fairly quickly noticed that most authentication protocols assume verification has already been done by means of an existing PKI, for example, which has already identified all the partners. So when you make naming issues available in their full glory you will see that this is not at all simple, and the Alice and Bob notation hides this because it's very convenient. We focused on other things, which is OK, but maintaining a PKI will, I guess, in most cases, overshadow all other costs in most systems, and we want to make this explicitly clear in the protocol, and not leave it, this is not just an implementation detail.

Now there are bad things about Alice and Bob, there are too many ambiguities, in particular related to the essence of how to deal with naming, but we're almost there, so Alice and Bob is a much better starting place.

Peter Ryan: There are things like Caspar which deliberately try to lay out the ambiguities.

Reply: Yes, we have hunted for an implementation which is claimed in two papers to exist, but we've not been able to obtain one, and we have searched quite hard.

Peter Ryan: Well I was thinking more of just a starting point for notation.

Reply: Yes, but we have come up with another notation for a different reason. They are on the right track, but we wanted to make sure that all our primitives were implementable, because we want to offer you a language where you can quickly type in your protocols and run them.

Ross Anderson: Why the name?

Reply: The language is named Obol, but although it's not spelt like this, Obol is a Roman denotation of the smallest weight that is available in the measuring system, so it's very, very small, and this is the purpose of the language. And it is efficient, measured in code size, it is, the implementation of protocols are double, more or less, of the size of a BAN protocol, which is very short, compared to a Java implementation; very short, so it's very efficient. It is not a general purpose programming language, you cannot implement quicksort. It is not a specification language, so it doesn't specify in the theoretical sense of specification, of course, an implementation is a specification, but it's not targeted at proofs, and it is not foolproof, it is awfully wrong to write incorrect programs, and this holds here as everywhere else. It is not a logic, it doesn't allow you to prove your program is correct, and it is not efficient measured in number of instructions.

George Danezis: You say it's inefficient in terms of instructions, but does that matter?

Reply: I mean, that it runs slow, if you write a two thousand line program in Obol, it will take forever, because it is a slow interpreted language. Another way to look at this, if you take the Alice and Bob notation, to do the BAN analysis, or any other analysis, you need the idealisation, OK, this is a manual process which is error prone, and you cannot prove this to be correct. The other way, from here to Java, you also need to do the same thing, it's called implementation. What we want to do is to close one of these gaps, to make something that can run as close as possible to Alice and Bob. We've not made it all the way, but rather close, you will see, therefore we don't care about efficiency, because it is only a small part of the system, and the ability to change, and to look at the security protocol and understand, is much more important than the performance.

Kenny Paterson: You mentioned earlier on in your talk that your approach would take care of the cryptography, so I have to ask what AES does this compiler use; for example, if it uses AES in a bad mode then a protocol that might look to be secure is actually totally insecure.

Reply: It is awfully wrong to write incorrect programs, if security properties of your protocol rely on that, then you must make this explicitly visible, which is the right thing to do.

Ross Anderson: There's also the attack on systems which goes, that someone says falsely that the protocol is broken when it isn't, and ships an upgrade which uses the old key material. So the new protocol does break and leaks the old key material, and so there have to be ways to control for that as well, upgradeability of protocols isn't necessarily a virtue, and reuse of crypto keys is usually bad.

Reply: Yes, this is true. Again, if you want to make sure that your crypto has specific properties, and you will notice that this format makes it easily extendable, you can add as many properties as you want.

Michael Roe: Can you explain the syntax where you've got star 1 and star 2?

Reply: OK, starred variables are variables where you have not made any explicit assumption on the type or the length, so you can not use them for anything except sending them on. If you want to use them for something, for example here, I could say, believe into variable X a shared key based on what I received, and then the run time can see, well this is only 10 bytes, I don't want to allow you, so this is not good. But when you're using it just to receive it and sending it on, they're typeless variables, because we think you need that.

Bruce Christianson: When you say block, do you mean you block until you have another go at receiving or do you just block?

Reply: No, this code will block until some message has arrived that is decryptable in one of these manners, which is what you want.

James Heather: The question is, why are you allowing flexibility of sending A and B in either order.

Reply: So, when they receive, can receive in any order, so the run time will look at the components and match them.

Bruce Christianson: Ah, the "or" in line 5 is at the meta level, not part of the language.

Reply: Yes, the red is part of the presentation, it's not part of the language, sorry, so you can do this, or you can do that, OK. Very good observation, thank you. Yes.

George Danezis: Compiling protocols of a magnitude, like you're doing, gives you also the opportunity to tweak all the hash functions, and whatever crypto you have, in order to have unique operations at each stage, unless they need to match. This is something that traditionally we don't do when we present protocols, which I think happened initially because the protocols are too long, they don't fit anymore on one side of A4. But that has become a religious principle that says, you should really minimise, and understand precisely why you're sending everything you're sending. But it seems that protocols are actually more robust if you just include all the information you can think you might ever want to include, and tweak everything, so that every hash function is different. Have you thought about that at all?

Reply: No, but I see this fits in the line that we're thinking. The receive has rather a complicated machinery, because the system has a pool of incoming messages, and a pool of requests for processes waiting for things, and the run time looks at the messages. I know this is very complex, and that it's exponential for large messages, and so on, but the matching is flexible, so you can stick in code to do whatever you want in your system, without having to break the other side.

Bruce Christianson: Just to follow George's point, one reason for putting a hash of all the local state in, is to force the other side to do the checking that they should do. In effect saying, I know you're lazy, and you won't do the checking if you don't need to, so if you don't do the checking, you won't know the hash value, and you can't decrypt this message.

Reply: Yes, so this is OK, if you want your protocol to be like that, you can implement this, yes, of course.

George Danezis: But it can also be implemented transparently?

Reply: Yes, yes, yes.

Bruce Christianson: Which is the interesting thing, yes.

Michael Roe: The implication is that decrypt can happen in any order, so if you got a message in which the nonces were swapped around you would automatically pick it up?

Reply: Yes.

Michael Roe: Now that's the right thing to do if you just receive messages in clear text and can swap things round anyway, but for the stuff that's protected by encryption, there are sometimes protocols where the correctness of the protocol relies on the fact that the attacker can't swap things round. That's why the mode of AES matters: there are some block chaining modes where the attacker can swap things round without needing to decrypt, and there are others where he can't, and it sometimes matters for the correctness of the protocol. So I'm a bit worried that you've built in that definition of decrypt.

Reply: Yes, we've thought about that; and our answer is that we believe you should not rely on very low level issues like this. This is an ongoing project, but we think this is the wrong thing.

Alex Shafarenko: What if a message doesn't match what you have?

Reply: Incoming messages are pooled, waiting for a reply.

Alex Shafarenko: Yes, but if it never matches the formats that you expect, it looks as though you always ignore that message.

Reply: Yes, the run time would after a while throw it away.

Alex Shafarenko: What if this is a sign of some form of attack and you want to notify the police?

Reply: OK. Now a part of what we want to do with this system is to have a system wide pool of incoming messages, just for this purpose, so that it's much easier to analyse the whole set of messages that comes in. For this purpose, you want them to pool up so that you can look at them outside the application. The system maintenance system is able to monitor the flow of messages because they're all passed through the same pool, and if they pile up, for example, this is a clear sign that something is wrong at the other end, which you don't want to embed inside the application.

Bruce Christianson: Now that's a very good point, and there are many cases like that. But in other cases I do want, within the protocol, to say, I want a message which matches on fields 1, 3 and 5, and *then* I want to check field 2.

Reply: And this is precisely what you do here, you return the fields that you don't match, either because you don't want to, or you don't know how to. You return them, and then you can do things with these fields.

Bruce Christianson: And one of the things I might want to do is to compare it.

Reply: Precisely, and note down that this didn't match what I expected, for example. Now, as we know, all problems in computer science can be solved by adding a level of indirection[1], and in some senses, using Obol is a level of indirection, because rather than implementing in some concrete style, Obol is interpreted, thereby giving a level of flexibility which we think is needed for security protocols.

Now the fact that the protocol is so small means that it's possible, for example, that you can fax a protocol to somebody. If a PKI, for example, fails, you can

[1] LNCS 3957 p324.

say, OK, we need to change from public key to shared key, now, I'll fax you the new protocol, type it in, because we need to get on the air, you can say this is far fetched but this is actually possible. There is room easily on a smartcard for a protocol, and there is room, without any problem, within an X.509 certificate for a full protocol, not just the key on the signed hash of an implementation in a .jar file that exists somewhere else relying on a PKI - there is room for the protocol itself.

Ross Anderson: I like the idea of putting a protocol in the certificate that certifies the key.

Reply: Because then the protocol can be revoked.

Ross Anderson: But I don't like the idea of people being able to send out protocols with their phish.

Reply: Of course, again, powerful tools makes for powerful damage, so this is a threat, of course, but we think that new possibilities become possible. With Obol you can say that we'll leave implementation as an exercise for the reader, which earlier was a no-go, but now it has become possible because the reader can implement the protocol himself. And of course, this is good for students (or bad), because now they can be given assignments, actually implement protocols.

Now we return to the order of elements. Interesting question for which heated debate often arises is, are these two messages the same, or not. Well, the answer is, of course, that depends. The bit strings are obviously different, but we know that all encrypted messages with different initialisation vectors are different, so at a logical level you care only about the content, the rest is engineering.

Chris Mitchell: But if you don't check the order, I'll send you an 8-bit ASCII character with three ones and five zeros in it, if you let me move the ones around you get a different character, the meaning changes, so ordering is essential.

Reply: At a very low implementation level, it is, but...

Chris Mitchell: But at all levels, if I reorder the letters in English, I get a different word.

Alex Shafarenko: You could label every component. Don't have a natural order, you can order them, but they don't have a natural order, it's effectively a function.

Reply: Precisely

James Heather: But you're throwing away many of the techniques that we've built up over the last ten years or so about verifying protocols, because your implementation might well make something insecure that has been previously secure under the assumptions that are more usually made about not reordering everything.

Reply: Yes, this is true. On the other hand you are saying that these two messages are different from a security point of view.

George Danezis: I can think of quite a few protocols where this is the case, where, for example, the order denotes your understanding of whether you're the boy or the girl in a protocol relationship.

Reply: My counter-argument to this is good engineering, and not trying to save bytes by not having redundant information inside the messages to guard against these things.

George Danezis: No, but it's perfectly secure. It's not an insecure protocol, it might not be robust for someone who can assume that he can reorder the stuff, but as it stands, it is secure.

Reply: I think that if your protocol demands that this comes before that, then that should be embedded visibly somewhere at this level.

George Danezis: Well, yes, but your notation makes it very explicit, and very visible, *i.e.* n_A, n_B, and comma usually denotes sequences, right.

Reply: Or sets. [Laughter]

James Heather: That means if n_A and n_B are the same, then you're only filling one thing out, right, because it's a set?

Reply: If the two nonces are the same then you have other problems.

James Heather: Well only if the protocol designers told you that you shouldn't have, I think that's the point: it depends on what your protocol specifier says that the protocol should you do, and if he said, I want these to be in this order, then it has to go within that order, if he hasn't said, these two answers have got to be different, then it's OK for them to be the same.

Reply: Well, OK. But in the real world, you cannot decide something about someone else. I send you a message, and we agree that you send me a nonce, if you do something odd, I have a problem, and I cannot specify that away, I need to deal with it. If the two nonces are the same, my run time will say, well I found your nonce here, but I didn't find something else, there seems to be something wrong, which, first of all, is the correct thing, secondly, it means that someone is trying to to do something which they shouldn't do, and I need the run time to deal with it, I cannot specify that there will be no dishonest party, and therefore not having the redundancy to deal with it.

James Heather: Well what you've done now is to change the protocol. You've put another nonce protocol rule that wasn't there in what was specified, and really that's a design issue, and it's down to the designer of the protocol whether those two nonces should be allowed to be the same or not. I mean, are you going to rule out A and B being the same, because I can think of situations where you might want a machine to talk to itself?

Reply: This is OK, but then that must be explicitly visible in your code that we allow this to be the same, because, as I said, naming issues are crucial for security, and when you implement a protocol you should make these things visible.

Bruno Crispo: It's a fair point, and essentially if you order a message you should specify it.

Reply: Yes, I think so, I think this is the correct thing, as I said, this normally ends up in a heated debate, I did tell you.

Wenbo Mao: To answer whether these two are the same, requires the protocol partners to check the semantics, that would be better, and also be more clear. If you just say it is protocol, then you haven't checked redundancy whether the received message is valid or garbled. You have to check it, if your protocol doesn't require this checking, I can see then no difference, they're all garbled, otherwise they have different semantics.

Reply: OK, yes, so let me make another observation, at what level are these different? Assume for one moment that implementation makes sure that you don't confuse them, then from a security point of view you have received the same information, regardless of the order.

Alex Shafarenko: This seems to be a language design issue, not a security issue, and at language conferences, what we usually say in these circumstances is that semantically it's there, because you can include a selector, which tells you in which sequence the fields are coming. If you're arguing about the syntax, go away and implement and your macro which actually gives you the syntax that you like. So it's not a security issue at all.

Reply: Yes, precisely, so, and our claim is that with security protocols, if order is important, this should be included in the protocol itself, it should not be left as a footnote to remember that these must never be swapped.

Wenbo Mao: So, the language everybody uses to send Bob something, you think that is not precise enough? It's all things to all people, but your language will become more specific by adding more information?

Reply: Yes, removing, ambiguities makes it less beautiful, but means we can execute it directly.

Alex Shafarenko: Is there any functionality for cryptography, or is it just protocol, with long numbers?

Reply: You explicitly say what encrypt and decrypt is supposed to be.

James Malcolm: Can you write your own encryption or decryption?

Reply: Yes, you can plug-in whatever you want.

Alex Shafarenko: In a different language, or in the same language?

Per Harald Myrvang: You use the run time implementation language, which is Java.

Alex Shafarenko: So if you want to use a different crypto scheme then you program it in Java, and plug it into Obol, you don't write it in Obol itself?

Reply: No, Obol is just for protocols.

Matt Blaze: Could I write it in Obol? I mean is this a Turing complete language?

Reply: No, I don't think so. I didn't want to bring this up, but for example in Needham-Schroeder you need $N + 1$ and since we don't want to support arithmetic in the language you have to call this outside. So if you have elements you need to use, we have support, but this again adds ugliness. You can evaluate according to some specification, and see is this true or not.

Alex Shafarenko: Why not have support for long numbers and the other things, in Obol direct.

Reply: Part of what we're trying to do is to restrain ourselves. I didn't show you everything in the language now, because there are more things under the hood available to you that I didn't want to present there, because they add also complexity, and they also add the possibility of errors. We're trying to find the trade-off, it seems that protocols that use these things are also hard to make right.

Alex Shafarenko: The problem with that is that you make this protocol less self-contained than it might be, because AES, for instance, is a reference to something defined elsewhere, and if you are to ship this protocol to a client, then you need to supply all the bits, and make sure that the version of AES, matches the version of your protocol in Obol. The supporting library is a logistical problem.

Bruce Christianson: Although to be fair he's not doing any worse than anybody else. [Laughter]

Reply: The problem is severe. On the other hand, you lessen that by not, for example, demanding it be AES. This is, we believe, implementation detail.

Ross Anderson: Well the things like algorithm negotiation failures are part of the problem. That's not an implementation detail, that's security protocols.

Kenny Paterson: If you look at something like SSL, or IKE, they spend an enormous amount of time making sure that you negotiate compatibility — that's actually part of the protocol.

Bruce Christianson: There are two agendas here, the first is, this nice language Obol which allows you to implement and specify these protocols in a particular way. The second point is that obviously you only want this weapon to be used for good, and so you want to put restrictions on it that protect people from their baser instincts. I think perhaps we should separate discussion about which things are bad, from the discussion about the architecture.

Reply: Yes, if for some reason, you don't like MD5 anymore, you don't want the language to force you to use it.

George Danezis: The last ten years have re-invented the idea of a sub-routine as well, because protocols are getting a bit too big, and it's quite handy to say, OK, Alice and Bob first run a verification protocol, and then they move to this other protocol. Do you have support for that kind of stuff?

Reply: Yes, run something else, and return here afterwards. But there are two problems, first of all error handling becomes rather hard, secondly, as we know, with compositional protocols, if you want to convince yourself that the combination of two protocols is as safe as the protocols themselves, this is very hard. For example, if you return a session key, will that be used for bulk encryption, or for something else?

Alex Shafarenko: OK, you have a type system here.

Reply: Yes. Here I assign, this variable has type.

Alex Shafarenko: That automatically means that any plug-ins that you use with this language, which would be written in Java, etc, should be equipped with a type signature as well, to match that type signature with whatever is required. So did you have a defined interface for plug-ins to allow an external plug-in to be equipped with a type signature, where does it come from?

Reply: Hardly. We have thought about typing in a more consistent manner, but this is very hard to do, and again, the question is, how much do you want to add to the complexity, both in the implementation, and also the size of the message, because then you need to take care of more things as you said.

Kenny Paterson: What about things that are open to interpretation and there's these two incompatible implementations?

Reply: There's only one way to see if you've done it correctly, to test in the existing implementation. So implementations are not defined by standards, they're defined by other implementations, and we know this very well. Although, in theory not. I've been to many IPSec conferences where one of the hot issues is that now we can speak to you, when we couldn't last time. We think that because we have a well-defined way of mapping it means that you can start by writing in Obol compatible hardware. You don't need to run Obol, but you need, if you receive messages, to behave compatibly, and from there you can develop the implementation further. So we think very modestly, and if the world changed to this it would be a better place.

Srijith Nair: How big is your Obol interpreter?

Reply: About 17,000 lines I think, including comments.

Srijith Nair: One of the uses you said is to ship the protocol inside a chip, so where is the interpreter in that case?

Reply: All smartcards have shared key encryptions, that means you can only interpret to someone else, someone you can trust at home, you can ship the protocol to them and get a result back.

Wenbo Mao: Subtle (or maybe stupid) protocols are broken, not because of the algorithm you use; the protocol is flawed because of some stupid or subtle design error hidden within it, not because of language imprecision.

Reply: If I can just quote Roger Needham on this, it is awfully wrong to write incorrect programs, and we believe that protocols in Obol are less hard to debug than their Java counterparts, basically because they're shorter. If you make very complex protocols, they become very complex also in Obol, obviously, because inherent complexity cannot be hidden. The good thing is that, you see this happening immediately because the programs will expand. So you say, this is getting out of hand, are we sure that we really need this? Obol is not a tool for everybody in all possible situations, obviously, but it replaces the Alice and Bob notation, both in teaching, and experimentation; so you can easily mock up a protocol, use an existing protocol, plug-in to your system, and you're up and running, so that you can deal with side effects. The day you have nothing else to do, you can go back and write 20,000 lines of Java to implement it properly.

Bruce Christianson: I really like the idea that you can write a specification, and that specification can be animated, and I'm not bothered about the fact that it's interpreted, because if you've got an interpreter interpreting, you can get a compiled version relatively straightforwardly by partial evaluation[2]. But another question is, can you, by annotating the beliefs, get a set of hints to some kind of reasoning engine that would allow you to use this specification as a basis for proving properties, as well as for implementing?

Reply: This I don't know, it is not my field.

Tuomas Aura: In the Security Foundations Workshop this year there's a paper by Andy Gordon and others. They write their protocols in a functional language called F#, and then they both execute and verify.

Bruce Christianson: Yes, that's the kind of thing.

Reply: But then they need to run on both sides I guess.

Tuomas Aura: No I think they are XML based protocols, so that's why they can actually improvise with the other side.

Bruce Christianson: OK, you only have to assume that the other side is consistent with the specification.

Alex Shafarenko: You're saying that this language doesn't need much arithmetic, etc, because it's all plugged in, but what if you have a protocol where you have a hundred bits, and then you burn five, or ten bits, right, by just extracting these five bits and sending. So you need at least a primitive that takes five bits out of a 100, and then five and 100 could be parameters, and I can imagine a protocol where you would also need a figure of four and 99 at some point. So you do need some manipulation.

Reply: Yes, this is true, and you will notice we've chosen the syntax so that it is very simple. But every security protocol needs a prior protocol for determining the language that we're going to speak.

Mike Bond: It seems that you've abstracted quite a lot away, aspects of making messages which would make it difficult for me to understand compatibility. For instance, if I wanted to get two identical protocols and make them physically incompatible, how would I do that?

Reply: Add an extra field, and then the protocol waiting for a message with four fields if it sees five will say: I don't know what this is.

Mike Bond: But can you only do it on a number of fields?

Per Harald Myrvang: No, you can use this on the content of fields as well.

[2] N. D. Jones, *"An Introduction to Partial Evaluation"*, ACM Computing Surveys, 1996.

Privacy-Sensitive Congestion Charging

Alastair R. Beresford, Jonathan J. Davies, and Robert K. Harle

Computer Laboratory,
University of Cambridge,
15 J.J. Thomson Avenue, Cambridge, UK, CB3 0FD
{arb33,jjd27,rkh23}@cam.ac.uk

Abstract. National-scale congestion charging schemes are increasingly viewed as the most viable long-term strategy for controlling congestion and maintaining the viability of the road network. In this paper we challenge the widely held belief that enforceable and economically viable congestion charging schemes require drivers to give up their location privacy to the government. Instead we explore an alternative scheme where privately-owned cars enforce congestion charge payments by using an on-board vehicle unit containing a camera and wireless communications. Our solution prevents centralised tracking of vehicle movements but raises an important issue: should we trust our neighbours with a little personal information in preference to entrusting it all to the government?

A 2003 study into the efficiency of transport [9] determined that the cost of traffic congestion to the United Kingdom economy is nearly £15bn p.a. (1998 prices), which constitutes 1.5% of the nation's GDP. Furthermore, the RAC Foundation claim that this is set to double within the next decade [11].

The UK Government has recently proposed [8,4] the implementation of a nationwide congestion charging scheme and has conducted a study into its feasibility [3]. The study suggests that such a scheme will encourage people to consider more carefully how and when they travel, and will provide incentives for them to travel at off-peak times, thus reducing the peak volume of traffic on the roads.

Traditional congestion charging schemes have employed either toll booths accepting cash payment, or vehicle-mounted tags which identify pre-paid accounts, interrogated by custom systems installed on overhead gantries. Whilst both of these systems have attractive properties in terms of maintaining the privacy of their users, they cannot scale to a national level due to the high cost of their installation and maintenance. Similarly, camera-based systems like those deployed in London would be prohibitively costly to extend to all roads nationwide. An alternative is for vehicles to contain on-board units which automatically calculate the congestion charge and decrement an on-board balance, but similar systems have, in the past, proven difficult to make tamperproof. To avoid this potential pitfall, an on-board unit could regularly upload the host vehicle's location to the congestion charging authority who then calculates the appropriate charge and issues a bill. This approach has particularly undesirable properties with respect to privacy.

In this paper, we propose a novel congestion charging scheme which, in our view, increases the privacy of its users, whilst still ensuring that enforcement of payment is possible. The scheme is particularly interesting because the vehicles form an important component in enforcement: without the support of the majority of drivers, its effectiveness would be severely diminished.

1 Threat Model

There are three types of entity taking part in our protocol: vehicle units, payment authorities and enforcement agencies. Vehicle units consist of an outward-facing video camera, a short-range communications unit, a location sensor such as GPS, and a microprocessor. Payment authorities collect congestion charge payments from vehicle units and issue a signed digital certificate to the vehicle unit for any payment made. Enforcement agencies collect data from vehicle units and use this data to issue penalties for non-payment.

We assume that vehicle units have been programmed with the correct public key for each payment authority and enforcement authority. Further, we assume that digital certificates issued by any payment authority cannot be forged, and that the enforcement agencies trust the contents of any valid digital certificate signed by a payment authority. Finally, we assume that vehicle units have some way of communicating with other nearby vehicle units as well as with payment authorities and enforcement agencies. We do *not* assume that the equipment installed in vehicles or their number plates are tamperproof. Therefore vehicle owners may attempt to modify any vehicle under their control.

By assuming vehicle units can be tampered with, we reduce the implementation complexity of the units at the expense of the need for greater care in protocol design. The overall aim of the protocol is to ensure:

– *location privacy for honest users*: the whereabouts of individuals who have paid the congestion charge are not recorded by either the payment authority or enforcement agency; and
– *fare-dodgers are detected*: individuals who have not paid the congestion charge are tracked and sufficient evidence is collected to ensure an enforcement authority can invoke penalties for non-payment.

We will evaluate our proposal in light of these two requirements later.

2 Protocol Description

The overall aim of the protocol is to allow the vehicle units to collect evidence of fare-dodgers and provide this information to the enforcement authorities. By doing this, movement data is processed in a decentralised fashion and therefore the centralised collection of location data concerning honest drivers is minimised. Our proposed protocol can be separated into three main phases: payment, usage and enforcement. We will examine these three areas in turn.

2.1 Payment

A vehicle owner who wishes to make use of the public roads must pay a fee. The fee can be based on any number of criteria, including the time of day, type of road, etc. In order to preserve privacy, payments are made through an anonymous payment scheme such as digital cash [2], and in return the driver receives a digital certificate signed by a payment authority. There may be more than one payment authority, but only one payment authority can operate in a single geographical region.

Every payment authority segments all chargeable roads for its region into spatio-temporal units with a single fixed price; we call such units *chargeable zones*. The set of chargeable zones must be communicated to all vehicle units to enable effective enforcement. Example chargeable zones for a payment authority may include £1 for use of the M11 from Junction 10 to Junction 11 between 8am and 9am, and £1 for use of the M11 from Junction 9 to Junction 14 between 10am and 11am. By ensuring all zones have the same cost, the payment authority can sign journey details received from vehicle owners in return for payment *without needing to see the journey details*. Roads with higher potential levels of congestion would tend to have many more (shorter) zones so that the total cost of travelling along the road is higher. By using a blind signature scheme the vehicle owner can receive a digital certificate (containing details of the road segment, time of travel and vehicle registration plate number) signed by the payment authority, whilst maintaining location privacy. More formally, for a vehicle unit V, with registration plate R, wanting to purchase travel in zone Z from payment authority P, with digital cash D, the protocol operates as follows:

$$\begin{aligned} V \rightarrow P &: \{\{Z, R\}_{K_V}, D\}_{K_P} \\ P \rightarrow V &: \{\{Z, R\}_{K_V}\}_{K_P^{-1}} \end{aligned}$$

If K_V and K_P^{-1} are commutative functions, V can retrieve $\{Z, R\}_{K_P^{-1}}$. The vehicle unit can present the certificate $\{Z, R\}_{K_P^{-1}}$ as proof of payment to anyone who requires it.

2.2 Usage

When travelling on a chargeable road, one vehicle unit, A, may encounter another vehicle unit, B. Our protocol requires some method by which A can determine the identity of B. For example, the registration plate of B may fall into the field of view of the on-board camera of A. In this scenario, unit A uses automatic number-plate recognition software to identify the presence and registration plate R_B of B, and a still photograph of B is taken using the camera on A's vehicle unit and stored in temporary storage. The time of the sighting and the location of A is derived from the unit's location sensor, and is also recorded with the photograph. Then A, using its radio communications module, challenges B for a valid digital certificate which proves that B has paid the relevant congestion charge. Note that a ubiquitous radio network is not required for this—the two vehicles simply need to communicate directly with each other. Either one of two outcomes occurs:

– B responds by presenting a valid digital certificate $\{Z, R_B\}_{K_P^{-1}}$ to A, in which case A deduces that B is entitled to use the road, and so deletes the photograph of B and takes no further action; or
– B responds with an invalid digital certificate, or B does not respond at all, in which cases A stores the photo of B and the associated time and location in permanent storage.

2.3 Enforcement

Enforcement is carried out by one or more enforcement agencies. At a convenient moment when vehicle unit A receives a network connection via its radio module, A uploads all photos of vehicles which responded with invalid digital certificates, or did not respond at all, to an enforcement agency. This upload does not have to happen immediately, but a fairly prompt delivery of data (of the order of days) is useful for the final part of the protocol.

Every enforcement agency aggregates the reports received from vehicle units at regular (say weekly) intervals. If a particular vehicle has been reported a large number of times, this may indicate that the vehicle was travelling without paying the congestion charge. If the vehicle has indeed been travelling in that location but has paid the congestion charge, the vehicle owner can present the relevant digital certificates to prove payment. In other words, if a vehicle with registration plate R has been spotted travelling in zones Z_1, Z_2 and Z_3, the vehicle's owner should be able to present $\{Z_1, R\}_{K_A^{-1}}$, $\{Z_2, R\}_{K_A^{-1}}$, $\{Z_3, R\}_{K_A^{-1}}$.

In general we wish to avoid such false positives since they waste time and result in reduced location privacy for drivers. False negatives are also troublesome, since these occur when fare-dodgers travel for free, and remove the economic incentives intended to moderate road usage. But, if we assume that all vehicle units operate faithfully, the probability that a fare-dodger will be caught for non-payment is strongly positively correlated with the volume of traffic; thus, when the charge is highest, offenders are most likely to be detected. However, in our threat model we assumed that vehicle number plates and vehicle units are not tamperproof. Therefore unscrupulous individuals may attempt to modify either their registration plate or vehicle unit.

If a vehicle registration plate is modified, replaced or removed, determining the real identity of the vehicle is very difficult. Thus, using the registration plate in a congestion charging scheme provides an additional incentive for tampering with it and therefore we expect the frequency of this crime to increase. However, assuming the deployment of a national-scale scheme with a reasonable proportion of vehicles equipped with vehicle units, the enforcement agency may be able to track the location and whereabouts of vehicles who do not have valid registration plates (or have none at all) by using vehicle unit cameras to track the movement of objects which are likely to represent cars. Such information may help the police trace vehicles without valid license plates and perhaps also the driver. Our scheme is open to one number plate modification attack: two or more vehicles can use the same registration plate in order to pay a single fee for the congestion charge. Such vehicles would then have to travel substantially along the same route at the same time if a moderate amount of saving is to be obtained. If two cars with the same number plate travel along different routes, then separate payments would have to be made in order to avoid an enforcement action.

Tampering with the vehicle unit itself leads to several further problems. Firstly an attacker can attempt to prevent one or more sensors on the nearby vehicle units from functioning. Examples include shining light at nearby cameras in order to prevent capture of photographic evidence [12], and jamming or altering GPS radio transmissions to prevent adjacent vehicle units from determining their correct current location. Communication interfaces to and from the vehicle unit could also be compromised. For example, care is required when a vehicle unit talks to the enforcement agency to ensure that the data is correctly transmitted: a man-in-the-middle attack may not be able to read the communication if data is encrypted with the public key of the authority, but a failure to forward all the relevant data must be detected and retransmitted, perhaps at a later point in time.

The enforcement agency may also receive erroneous data from modified vehicle units. For example, a malicious individual may upload a forged photograph, or attach an incorrect charging zone to a genuine photo. Such a modified submission may then lend credence to a vehicle owner being falsely accused of using a particular charging zone. Since the vehicle owner may have paid the congestion charge and driven elsewhere or even left the vehicle in a garage there will, in many cases, be no record of the vehicle's actual movements.

One solution to this problem is to attempt to detect the fake evidence and punish those who submit it. Digital watermarking of photographs is not suitable, since the vehicle unit is in the hands of a malicious user who may modify the vehicle unit to apply the watermark to the fake photograph. Statistical analysis of the digital image may help to indicate whether an image has been forged, even if it appears genuine to the human eye [10]. However, such techniques do not protect against forged charging zones.

We require better protection against forged evidence. The next two subsections explore whether we can protect against forgeries whilst still permitting anonymous reporting, or whether reports should only be accepted from certified entities.

Anonymous Reporting. If enforcement agencies must rely on anonymous reports, forged data may be detected by examining evidence from many distinct vehicle units. This approach assumes that wide-spread collusion is not possible. Using this approach, we require a method to ensure enforcement data has originated from different vehicles without compromising the identity of the vehicle's driver. One solution is to issue a unique public/private key pair with each vehicle unit, and to sign the public key using the private key of a payment authority. We can then use this certified vehicle unit key when data is submitted to the enforcement agency. More formally, for a photograph G of a car with registration plate R, taken in charging zone Z, the vehicle unit A (purchased from payment authority P) is equipped with public/private key pair K_A/K_A^{-1} and can submit the following evidence to an enforcement agency E:

$$A \rightarrow E : \{K_A\}_{K_P^{-1}}, \{G, R, Z\}_{K_A^{-1}}$$

By validating the key used by the vehicle unit, an enforcement agency can ensure that data used to determine whether a vehicle is travelling without paying is collected from many distinct vehicle units. Data from vehicle units which have in the past submitted evidence of questionable integrity can be ignored by the enforcement agency. If the reported data ($\{K_A\}_{K_P^{-1}}, \{G, R, Z\}_{K_A^{-1}}$) is found to be questionable, the offending vehicle unit keys can be communicated to payment authorities. The payment authorities can then prevent the offending vehicle unit from purchasing new congestion charging credit assuming, of course, that the payment protocol is modified so that keys must be presented to purchase congestion credit. An indirect fine for submitting fake evidence can then be imposed by charging for the issuance of a new public/private key pair.

Unfortunately such a scheme can lead to invasions of location privacy. The key pair represents a static pseudonym for the vehicle unit, and this enables the enforcement agency to correlate together all the places that the vehicle unit has reported potential offenders. Therefore charging zones must be large enough to prevent any zone from having a strong sense of identity attached to it. For example, we would not allow a section of road connecting a single house to the wider road network to act as an individual charging zone.

Even with this precaution, some privacy violations may continue to exist. For example, if charging zones were created at the granularity of postcodes or zipcodes, then the combination of reports from several zones may uniquely identify an individual. This is perhaps more likely if the released zones include those at the start and end of a long journey. The privacy risks posed can be reduced by limiting the number of reports and exchanging vehicle units (and therefore keys) at regular intervals. Interestingly, the greater the number of fare-dodgers, the less location privacy is afforded to honest participants who file reports with the enforcement agency.

In Section 2.1 we described how a blind signature scheme is used to prevent the payment authority from receiving any journey information associated with a purchase. This means the payment authority (unlike the enforcement agency) can only correlate financial expenditure and time of purchases with the vehicle unit public/private key pair. Intuitively, this appears to pose less risk of re-identification for the owner of a vehicle unit.

Certified Reporting. Our solution for anonymous reporting assumes that wide-spread collusion between vehicle owners was not possible. An alternative scheme which does not rely on this assumption ties any report of bad behaviour to an explicit identity, and offers citizens the chance to submit a traceable report. This route to evidence collection may result in less reports, but allows a more traditional form of witness statement to be collected from a legal entity. This solution allows visual inspection of the vehicle unit to check for tampering—the usefulness of this approach depends on whether a tamper-evident package is easier to build than a tamperproof one.

It is also possible to make two versions of the vehicle unit: one with enforcement potential (*i.e.* with a camera) and one without. Less comprehensive enforcement may then be possible by only installing the enforcement version on public service vehicles. All vehicles still require a vehicle unit to confirm payment of the congestion charge and therefore maintain location privacy.

3 Discussion

Our congestion charging protocol places some trust in vehicle units to execute the protocol faithfully. If very few vehicle units function correctly, very little enforcement can be carried out. To be truly effective, the charging mechanism requires a critical mass of vehicles to support the scheme. In this paper we have only discussed the validation of congestion charge payments. However, the same scheme can also be used to check for valid digital equivalents of vehicle roadworthiness certificates, general road-tax payments and vehicle insurance.

We believe the system does achieve a greater overall degree of privacy than a centralised system which monitors the movement of all vehicles—our scheme only reveals movement data for those vehicles who have not paid, or do not respond when challenged. This provides an incentive for vehicle owners to ensure that their on-board units are operating correctly and discourages destructive tampering.

Road users would prefer neither the government nor other road users spy on them. However, assuming that a national congestion charging scheme is essential for the future viability of the UK road network, which is the lesser of the two evils? It is our belief that the system we propose is more desirable than a centralised one. In earlier work [7], we gave an overview of this protocol as an example of why privacy issues require greater attention when designing systems. Most responses we have received have been positive, but some were clearly negative, for example:

> "*[W]asn't the last time a population was coerced into spying and reporting on its neighbours assigned to the history books with the disbandment of the Stasi following the collapse of East Germany in 1989?*" [1]

This viewpoint may, in part, be due to the lack of belief in the need for congestion charging. Alternatively, some individuals may prefer to trust the government with their location data, rather than trust their neighbour to provide fair enforcement.

It is important to note that most of the technology to build a vehicle unit exists already. Drivers can install cameras in their cars today and report illegal behaviour to the police. Nevertheless our proposal does encourage wide-scale surveillance and stream-lines the reporting mechanism.

In the UK at least, the general public has a long history of reporting wrongdoing to the state; for example, there are presently mechanisms for citizens to report benefit fraud [5] and unlicensed vehicles [6]. Perhaps the difference between our proposal and current methods is that filing a complaint is voluntary rather than automated by technology (although our proposed protocol could be adjusted to provide users with an option to select when to report offences).

In summary, we believe that this scheme is worthy of study, as it challenges the opinion that user privacy can only be achieved in a large-scale congestion charging system at great cost. The nature of the scheme presented here differs radically from other proposals, particularly in that it relies on its participants to perform the enforcement of payment. Integral societal involvement in the protocol means that the efficacy of the system would depend heavily on the users' attitude toward it.

References

1. Brown, S.: Feedback: Big brother is back. IEE Review 51(12), 6 (2005)
2. Chaum, D., Fiat, A., Naor, M.: Untraceable electronic cash. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 319–327. Springer, Heidelberg (1990)
3. Department for Transport. Feasibility study of road pricing in the UK (July 2004)
4. Department for Transport. The Government's response to the Transport Select Committee's report, Road Pricing: The Next Steps (July 2005)
5. Department for Work and Pensions. Website, Report a cheat online form (2006), https://secure.dwp.gov.uk/benefitfraud/
6. Driver and Vehicle Licensing Agency. Website, "Reporting of unlicensed vehicles on the public highway" (2006), http://www.dvla.gov.uk/public/unlic_veh/report_online.htm
7. Harle, R., Beresford, A.: Keeping big brother off the road. IEE Review 51(10), 34–37 (2005)
8. House of Commons Transport Committee. Road pricing: The next steps (March 2005)
9. May, A.D., Allsop, R.E., Andrews, D.J., Betts, C.V., Bayliss, D., Cottell, M.N.T., Dick, A.C., Kemp, R.J., Lowson, M.V., Ridley, T.M., Tietz, S.B., Wootton, H.J.: Transport 2050: The route to sustainable wealth creation. Technical report, Royal Academy of Engineering, 29 Great Peter Street, London, SW1P 3LW (March 2005)
10. Popescu, A.C., Farid, H.: Statistical tools for digital forensics. In: 6th International Workshop on Information Hiding, Toronto, Canada (2004)
11. RAC Foundation. New government—new agenda for transport?, (April 2005), http://www.racfoundation.org/index.php?option=com_content&task=view&id=%66&Itemid=35
12. Truong, K.N., Patel, S.N., Summet, J., Abowd, G.D.: Preventing camera recording by designing a capture-resistant environment. In: Beigl, M., Intille, S.S., Rekimoto, J., Tokuda, H. (eds.) UbiComp 2005. LNCS, vol. 3660, pp. 73–86. Springer, Heidelberg (2005)

Privacy-Sensitive Congestion Charging
(Transcript of Discussion)

Alastair R. Beresford

University of Cambridge

Tuomas Aura: How do you measure the cost of congestion?

Reply: It's estimated by economists. Traffic congestion is estimated to cost the UK economy about £15bn, and this impact is going to increase. I'm not entirely sure on the details of how they've made that estimate, but you could imagine saying, how much extra journey time is involved in, for example, truck journeys, than otherwise would be if there was nobody else on the road, etc.

Bruno Crispo: Is this about the whole country?

Reply: This is about the whole country, so that's motorways, trunk roads, maybe even country roads, the whole road system. And if we're going to start covering large portions of the road network, then people in the small villages are worried that drivers are going to use their small lanes in order to avoid paying a fee, and so the push at the moment is for a pervasive coverage.

Bruno Crispo: Can you explain what it means for a road to be congested?

Reply: That's a good question, I guess everyone knows the extremes; when there's a clear road there clearly is no congestion, and when you're stuck in a traffic jam and only moving at five miles an hour on a motorway you know you're definitely in congestion. If you look at the number of vehicles travelling on a road per hour, and look at the speed that they travel at, there's a very sharp drop-off at a particular traffic volume; say on a motorway where the speed drops from 60 or 70 miles an hour, sharply down to 30 miles an hour. I guess you could say that the road has become congested at that point. And the whole aim of congestion charging is actually to prevent us from getting in that situation, so ironically you pay the congestion charge to avoid congestion, rather than because you're in congestion.

If a vehicle owner hasn't paid then we need to collect data that's strong enough to ensure some kind of conviction, or at least enforce some kind of penalty fine. If all the vehicles do operate faithfully though, you'll notice that the probability of catching someone that's trying to evade paying is very positively correlated with volume of traffic. Therefore, in an area where congestion charge is due, you're very likely to be caught, but in places where there is no traffic at all, then maybe you shouldn't have to pay congestion charges in the first place, so it doesn't really matter if you get away with that.

Frank Stajano: I'm not sure I entirely understand the second point, is this a sort of "cover your back" thing, maybe someone has filed a false report to try and frame you, they're saying that you haven't paid, but actually you have paid?

Why is it not sufficient to have just one report for a particular case that said, I didn't get something to verify, then it's avoidance of a congestion charge.

Reply: On the transition between zones, if car A is actually on the edge of one zone, and car B is on the edge of another zone, then maybe you would have some concern, and maybe you would also be concerned that a certain percentage of cars have been hacked open and false reports are being filed.

The major concern I would have would be whether the car is on the boundary of a zone or not, we could try and reduce that to zero, but yes, I'm also concerned about the fraud aspect, and probably dominantly with that point.

Peter Ryan: Are these certificates time limited in some way?

Reply: The zone that you buy for is both the spatial region and a temporal quantity, so you'd buy from 8 to 9 o'clock in the morning for a particular region.

Peter Ryan: OK, so it's folded into the zone concept.

Bruno Crispo: It seems massive collusion is not so unlikely. Essentially we are carrying a system that means we are working for the enforcement agency against our own interests by taking this picture, so it may be that we are really not motivated to make the system work?

Marios Andreou: We may, for example, paint over the camera.

Reply: Yes, you can stop your own camera taking photos, but the point is you've got to stop all the other cameras that are around you taking photos.

Bruno Crispo: Are we all motivated to actually stop the camera?

Reply: Well it depends, this comes back to the point about congestion charging, and whether we think there's going to be a problem in the future or not, and so if we're happy having congestion then maybe this is a good scheme, because people can just do a bit of civil disobedience and avoid paying.

Chris Mitchell: A slightly different question relating to your point about pseudonyms. It seems to me that by distributing this process you actually make it easier to monitor the location of vehicles. If you have a single trusted authority, cars will only respond to this trusted authority in a way that perhaps conceals their identity from everybody else, whereas if we're all talking to each other, we all find out pseudonyms from one another, so imagine a network of Mafia guys who want to track individual users, they could just put their cars out on the road and track them, because everybody has to talk to everybody else, whereas with a centralised system at least the Mafia guys don't have the right to interrogate me.

Frank Stajano: All the Mafia guys are doing in this case, is a challenge response based on having acquired the number plate. They could forget the protocol and just read the number plates.

Reply: Yes, I don't think there's any extra things that you get with this that you don't have already with people putting cameras in cars.

Chris Mitchell: With a centralised authority you're tracking people all the time but with this system you just take a picture, it's not a constant process.

Reply: Yes, we can't track pervasively like the centralised system could, say where has this car gone 24/7.

My real concern with the anonymous reporting is that I'm not sure whether this scheme really provides sufficient guarantees for a court of law to enforce a penalty fine. So another approach is to try and attach the current driver's identity to any report that's filed, and then you can use the traditional kind of human protocol of, you go to court and make a testimony about the accuracy of the data that's been collected. But of course that may heavily discourage the use of this scheme because people don't want to go to court to make that kind of testimony.

Mike Bond: How on earth would an average car driver do that, would this device beep when the guy in front is evading the congestion charge?

Reply: You could do that, or it could be enough that were you on that road, at that time, on that day.

Ross Anderson: Presumably you need some incentive, so some proportion of the fine has to go to the people who catch them. That works.

Reply: Yes, otherwise why would you bother. Then maybe there'd then become a secondhand black market in trying to go round the people that otherwise would be taking you to court.

Ross Anderson: And I suspect there might also be a market in avoiders' clubs, because once you have got a numberplate that uses LCDs to display the number you want, which is trivial technology, then of course people can pool identities. We might very well find there is an optimum size of an identity pool of people who all pretend to be the same car, and share a congestion charge.

Reply: It depends on the size of the zones, if you make the zones small enough then people will have to follow the same route at the same time, and then maybe they should just car-share anyway.

Ross Anderson: There's also the issue that once you get an LCD numberplate you can do middle person attacks on this.

Reply: Yes, show the numberplate of the car in front to the car behind.

Ross Anderson: And then he turns out to be a bad guy and you go to jail.

Reply: Potentially.

Frank Stajano: If you have an LCD numberplate you go to jail anyway.

Michael Roe: So, you would have a problem. London congestion pricing is already under the same attack.

Ross Anderson: It is a problem; if you own a blue Citroen and you live in London the trick is to find somebody who's got a blue Citroen, and a similar identity, happens all the time.

Reply: One solution for the certified reporting may be that we only install a camera enabled version in public service vehicles, and restrict testimony to come from those kind of people. At least I believe this system is better than a centralised one in the sense that the data that's collected is minimised, and it does require road users to cooperate in order to function, so coming back to your point, if society just prefers to sit in the congestion, well then they can.

Tuomas Aura: If you have satellite-based tracking of all vehicles, and you can collect this data, and you also need a system for locating the resource, and communicating with the central authority, then there's so much more you can do with that. For example, it's used for planning road building, you need to know where people came from and where they are going. It may be unpopular but it can be used for lots of useful things.

Reply: You mean the centralised scheme?

Tuomas Aura: Yes, but this kind of thing can't.

Reply: I guess you can still collect some data, so you could collect, for example, the number of cars on a particular segment of road, and then file that anonymously, so we could collect some useful statistics in the same way, but I'd have to think about what applications you wanted to do as to whether we can provide an adequate datafeed to it.

Perhaps a debate livener might be to say, well let's use this for speeding, there's no reason why you couldn't use the same scheme to try and catch photographs of people that speed, but...

Mike Bond: You'd have to be going at the same speed as them surely?

Reply: Well, similar but that would be an irony wouldn't it.

Michael Roe: If the zones are set appropriately, you could tell if someone was speeding by the rate at which they were requesting and buying signatures, and you know then, how many seconds between times going through zones.

Reply: You could prepay if you want to speed, I suppose, so that you could buy a set of zones ahead of time.

Tage Stabell-Kulø: If you want to experience average speed control you can just come to Norway. In our 20 kilometres zone, every car is photographed as you enter, you're photographed as you depart, and if it's too short you're fined.

Larry O'Gorman: A comment on the effectiveness of the camera: last year I was driving in London and I made a wrong turn and managed to end up in the congestion zone there, driving a rental car; a month and a half later I received a ticket, plus a charge from my car rental company. I write a regular article for a journal, and I mentioned this as an example of the effectiveness of OCR reading my plate there. But an engineer who worked for the congestion people wrote me an email about it saying that this is not the one of the wonders of OCR; they have a bank of people actually looking at the photographs reading the licence plates. So it's a difficult thing to do (and it was actually a good photograph of my car), when you've got this camera sitting on this car, it's bouncing up and down, and it's not aimed at the car in front, the pattern recognition is going to be a difficult thing to do.

Reply: Yes, so we're not hoping to get anywhere near the hundred percent reliability, but even if you got five or ten percent hits then it may still be sufficient.

Larry O'Gorman: Why don't you just use the signature from your RFID, I mean, each one has a transponder, and you're getting a signature?

Reply: The idea is to tie together some photographic evidence that may be permissible in court, so that was the aim of using the photograph.

Aaron Coble: Have you considered taking the Yahoo mail approach to pseudonymity, where you can get as many of these transponders as you want, you make them very cheap, and all you need is some ID associated with that transponder. You can preload money on it, and then you impose physical barriers, so you show up at an barrier and it reads your ID, records that, and whenever you want to leave a zone, you have to have enough money associated with that ID to leave it.

Reply: One of the big problems with using barrier based entry schemes for congestion charging is it causes a lot of congestion. If you look at things like the big bridges that are currently toll-based in the UK, there is a huge number of entry points to pay, and we haven't got space to build that kind of thing at 20 points around Cambridge city centre, so there are physical restrictions associated with doing that. But, in principle that would work.

Ross Anderson: Entry control is used in motorways, for example, in Los Angeles, where it's very effective, and the light goes green to allow one car through, then decides whether to wait two seconds, five seconds, or seven seconds, to let the next car in. If you're going to have a scheme like that, then you can combine it with coralling people on the motorway.

Bruno Crispo: But then you also need to stop.

Reply: Yes, with our scheme at least you don't have to stop anywhere, that's true. But if the road in Los Angeles is already at capacity, then I guess you might as well stop the people on entry rather than clogging up the entire roadway.

George Danezis: It's dangerous to have a distributed surveillance system, not only because it facilitates surveillance, but most importantly because it allows surveillance of things that are not intended. It's likely that somehow all these things, like peer networks, can be used for things that were not originally designed or intended. What do you think about that?

Reply: Yes, we may be concerned that, version 2 of the firmware that gets uploaded then does some things that we don't agree with. I guess one argument is that the boxes are still in the hands of the individuals and they can remove them, or spray over the camera, or whatever.

George Danezis: But it is exactly because they are in the hands of individuals, that the individuals can use them to serve their own goals, rather than the goals of Ken Livingstone[1]. The individuals are interested in knowing if there is a police patrol car waiting for them.

Reply: Oh, so you're suggesting the driver's actually pre-programmed the boxes to tell them or see from the camera five cars in front, or something like that. Yes, I guess that's possible. Maybe that's another good feature of the system, I don't know.

Ross Anderson: Maybe you can also listen to other people's tunes.

Mike Bond: There was a scheme running, I can't remember which country it was, possibly Germany, where a chap made a voluntary database of registrations and mobile phone numbers. That doesn't tell you outright if you get someone's mobile number, who they are, what their address is, but if they cut you up it allows you to ring them up, and hurl abuse at them.

[1] Mayor of London.

Chris Mitchell: It's not that hard to prevent downloading of software to a box. If you're manufacturing the box, you can control who gets the right to upgrade the firmware, and presumably if the government is providing these boxes for a specific purpose, they're not going to want them used for other purposes, especially if they're subsidising them, or producing them at very low cost by selling by the million. So you could just, for example, put a public key in, and require a signature on the code. OK, that doesn't stop the tampering with the boxes, but they can also buy their own boxes, so there's no real incentive to tamper with a box if it costs you $300 to re-jig the box, or maybe $100 for Mike.

Ross Anderson: Well you can't there, because BMW has the patent on the idea of code signing in road vehicles.

Chris Mitchell: I guess maybe we just wait till it runs out.

Ross Anderson: OK, but that won't be for 15 years.

Reply: My final point really is, who is big brother. There's been a lot of very supportive emails I've had back talking about the scheme[2], but also a couple of ones that are being quite derogatory, and so I wonder whether it really is that some people prefer to trust the Government than to trust a distributed scheme, or else people just don't believe that congestion charging is necessary.

[2] Some of them during this talk.

The Value of Location Information
A European-Wide Study

Dan Cvrcek[1], Marek Kumpost[1], Vashek Matyas[1], and George Danezis[2]

[1] Faculty of Informatics, Masaryk University, Brno
Botanicka 68a, CZ-602 00, Czech Republic
{cvrcek,xkumpost,matyas}@fi.muni.cz
[2] K.U. Leuven, ESAT/COSIC,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
George.Danezis@esat.kuleuven.be

Abstract. The value attached to privacy has become a common notion in the press, featuring frequent stories of people selling sensitive personal information for a couple of dollars. Syverson argues [1] that we should incorporate the risk of data misuse into our reasoning about privacy valuations. Yet there are doubts as to whether people can, and do, value their privacy correctly and appropriately.

Privacy is a complex notion and as such it is very difficult to valuate it taking into account its full complexity. In this experiment we consider one aspect of privacy, namely location privacy, that can be compromised through mobile phone network data. We performed a European-wide study to assess the value that people attach to their location privacy using tools from experimental psychology and economics. We present the first results here.

1 A Few Words at the Start

Privacy has always been an elusive security property. While it can be partly modelled as confidentiality, or controlled disclosure, of some personal information, individuals seem to be prepared to renounce it even for a modest reward – the popularity of store loyalty cards being a prime example. Similarly, despite declaring some sensitivity to their personal information being leaked, individuals seem to not be prepared to accept the overhead or cost of privacy enabled technologies. The market failures of flagship products like the Freedom network [2,3], an anonymous web browsing solution, illustrate this.

Yet more and more privacy intrusive technologies are deployed and become widely adopted. The GSM mobile network, to choose one that is considered indispensable in everyday life, allows for the tracking of powered-on handsets through the operator BTSs (Base Transceiver Station). This allows real-time tracking of mobile phone devices with a granularity of a few hundred meters within cities, to a few kilometres in less populated areas. The information is recorded by the operators, and often used for network management but also law enforcement operations. A private service is even available in the US that allows anyone to buy the call records associated to any mobile phone for about $160 [4].

It seems clear that the tools of computer security alone cannot give us the full picture of people's attitudes to privacy, and for this reason we turn to experimental economics to establish the "value" that individuals attach to their privacy. In particular, we shall generalise the study by Anderson *et al.* [5] on the value of location information – and perform it on a wider and more varied population and across multiple E.U. countries. Location is a quite relevant aspect as mobile phones are ubiquitous and can be used to eavesdrop on users' movements. Location privacy study by Anderson *et al.* was done in a relatively small scale at Cambridge University. By measuring the same aspects of privacy, we can compare our results, and establish whether people's attitudes to privacy are uniform across the EU.

Some studies about people's attitudes to privacy have already been published [6,5], and experimentally pinpointing the value people attach to privacy is a difficult problem. We chose, in the tradition of Anderson *et al.*, to use an auction, where people are required in effect to sell their private information. Participants have incentives to communicate to us the true value they attach to their privacy: bidding too high may exclude them from the study, while selling at a lower price than the minimum they are expecting would provide too much discomfort for them to participate.

The key ideas behind the auction methodology used are rather simple: it has been shown that, if they are asked directly in sociological studies, individuals tend to overemphasize how privacy sensitive they are – suggesting behavior that is generally not matched by their actions. For this reason we deceive the subjects of our study and make them believe they participate in an auction where they sell private location information (the consecutive mobile phone cells they used during a month) to use in a fictitious study on mobile phone usage. The lure of real financial returns, and the action structure of the study, gives all participants incentives to state their real privacy valuation on the disclosed location data.

Our study was conducted in the context of, and with the help, of the FIDIS[1] network. Partners in different European countries have been instrumental in allowing us to gather data across Europe.

2 Towards the New Study

The original study by Anderson *et al.* [5] that pioneered this approach was limited by several factors. Firstly the population on which it was carried out had very specific characteristics: computer science undergraduates at the University of Cambridge. These were mostly male, highly educated and technically aware. Although some conclusions were drawn about the correlation between privacy sensitivity and general use of the mobile phone, as well as patterns of travel, it was very difficult to assess correlations with gender or nationality. Secondly no

[1] FIDIS – "Future of Identity in the Information Society" is a 5-year Network of Excellence within the EU 6th Framework Program. Its objective is to research the changes that the concept of identity is undergoing in the developing European information society.

serious analysis was performed on the "self-selection" bias of the participants: those with high privacy sensitivity might have selected themselves out of the experiment, and not participated exactly for this reason. This could have lead to a constant bias in the study underestimating the value people attach to their private location data.

Our study attempts to remedy these shortcomings. We performed the same study *in multiple European countries, in which FIDIS institutions are based.* It allowed us to compare the results across different nationalities. Due to relatively large number of respondents, not only the median value attached to the data can be compared, insights into the privacy sensitivity of local populations are also provided by the overall distributions of valuations.

A second considerable improvement over the original experiment was the use of a more gender balanced population. Replies from such a population allow us to draw conclusions about a possible gender bias between men and women when it comes to the sensitivity of location information. Similarly our new study is performed on both Computer Science / Informatics as well as less technically savvy population, to draw better conclusions about the sensitivity of an average individual.

3 Implementing the Study

Our study involved a small amount of deception, namely that we informed participants that the study's object was a research on the mobile networks' structures. We considered this approach necessary to get less biased data. Despite deception being commonplace in such experiments, it has some ethical implications we are not used to. As a result, all participating institutions had to clear the experiment with their local ethics boards, and the participants had to be informed about the true purpose of the experiment immediately after it was finished.

The study was implemented using web forms containing a questionnaire. We advertised it using emails addressed to university mailing lists to involve as many people are possible. The text of emails also leaked to readers of a large mobile phone web server in the Czech Republic thanks to Czech students contributing to its content. The information text we used was as follows.

> Dear reader,
>
> <Institution> participates in a European-wide study organised within the FIDIS (Future of Identity in the Information Society – *www.fidis.net*). This study involves gathering location data for a number of volunteers over a period of 30 days.
>
> We are looking for people who will be monitored for the purpose of a sociological study into mobility of people and also with respect to the appropriateness of mobile phone network structures in regard to the requirements of mobile phone users. Please note that you should not switch off your mobile phone during this experiment.

The location data will be retained, and may be used again for future academic research. The location of mobile phones will be queried every 5 minutes 24x7 in cooperation with your mobile phone operator for the whole period of the study. The resolution of the position is within the phone network's "current cell" – about 800 meters in countryside, 100-200 meters in built-up areas. This querying will not be affecting any functions of mobile phones.

Each participant in the study will receive monetary compensation, and we are running an auction to select those who will take part. We invite you to submit a bid for the amount of money you require to take part in such a study. As our budget is fixed and limited, successful bidders will be those who bid the lowest amounts, and each will be paid the amount of compensation demanded by the lowest unsuccessful bidder.

Please visit the link www.buslab.org/FIDIS_experiment regardless of your intent to take (or not) part in this study.

Best regards,
<name of sender>

A similar introductory e-mail was sent out to university students in five countries (translated into the local languages and with the institution name of the local partner of the study).

3.1 Structuring the Questions

The web questionnaire was structured into four logical parts. The first part contained a longer version of the introductory email (very much similar to part 3.3 in [5]). We put a question about the respondent's interest in the study at the end of the text. There were three options a potential participant would choose from:

- I do not have a mobile phone
- I do have a mobile but I am not interested to participate
- I do have a mobile phone and I am interested to participate

Those answering they both have a mobile phone and are keen on taking part in this study, were presented with a request for their e-mail address. An email was then sent to them with a login name and a password for further communication through the web interface. After a successful enrollment and subsequent authentication, the following questions (with predefined options for answers 1-6 as pull-down menus) would be presented:

- Is your background (area of study)?
  a) Computer Science, Comp./El./Comm. Engineering, Informatics, b) Law, c) Other areas
- What is your gender?
- What network do you use for your main mobile phone?
  <list of local operators>

- Do you carry a mobile phone with you most of the time?
  a) yes, b) no
- How often do you make irregular movements (such as shopping, going out with a friend, pub, visiting friends)?
  a) several times a day, b) every day, c) every week, d) every month
- With whom do you communicate using your phone?
  a) friends, b) family, c) partner, d) business
- How much compensation would you require to participate in our study for 30 days (in whole amounts of local currency)?
- Text array for free comments.

The second form (a very short one) was presenting the subject with a possible change in the use of the collected data. The text stated that "there is now some possibility of commercial interest (from partners of your mobile phone operator) in the data collected during our study. We would be grateful if you could let us know whether you would"

- not be willing to participate in the study if the data might be used by a regular business partner of your mobile phone operator.
- allow your data to be used by a regular business partner of your mobile phone operator for the same amount of compensation you originally bid.
- allow your data to be used by a regular business partner of your mobile phone operator only if you were allowed to revise your bid for compensation.

The second form may have benefited from being presented to subjects a few days after the first one. This would allow the subject to "forget" their valuation, rather than facing a sharp contrast between "academic" use, and "commercial" use. This could have been done using the collected email addresses. We conjecture that the results would not have been significantly different, yet confirming this experimentally should be the subject of a further study.

The final form further modified the parameters of the study. It stated that "a business partner of your mobile phone operator inquired about the option to extend the experiment period to 12 months in total. Would you"

- decline your participation in the experiment.
- participate for the following amount of money <enter the sum>.

4 Basic Results of the Study

Our web questionnaires were up for about a month. We received majority of responses during the 48 hours after emails were sent out, except for the Slovak Republic where the information was being spread with paper notices. As a result, responses there were uniformly spread throughout the month of the experiment.

We present here the key features of the collected data. In particular, we describe the basic demographics and the distributions of the bids, by questions regarding gender and mobility. A more thorough analysis and modelling of the data is left for future work.

Table 1. Numbers of participants in countries

Country	Total	Women
Belgium	37	3
Czech Republic	744	131
Germany	251	33
Greece	30	6
Slovak Republic	152	46

Table 2. Who is being called by mobile phones

Calling	friend	family	partner	business
Number	1076	975	598	358

About 1200 participants answered the first set of questions. These were from five countries: Belgium, Czech Republic, Germany, Greece, and Slovak Republic – mainly representing people from the Central Europe, as well as from the Southern and the Western Europe. The partitioning of the participants according to their country of origin is in table 1. The sets of the Czech Republic, Germany, and the Slovak Republic are large enough to allow for very detailed structuring of results. The smaller sets of Belgium and Greece are used more as "control sets" to verify general results.

The set of the participants consists of 800 people with a background related to computers, 32 "lawyers", and 381 with another (unspecified) background. All but ten participants carry their mobile phone all the time.

The second demographic aspect concerns the frequency of irregular movements: 483 people do such movements several times a day, 520 about once a day, 194 in a week intervals and only 15 monthly – we again obtained three very large sets suitable for an analysis. Mobile phones are mostly used to catch up with friends. The second most often called on average is a family member, followed by a partner and business (table 2).

The first question to be answered was whether the respondents were interested in the study, not interested in the study, or whether they had no mobile phone. We call those who chose the second option "early drop-outs", and it was chosen by 11 people from Belgium, 85 from the Czech Republic, 65 from Germany, 32 from Greece, and 46 from the Slovak Republic. We recalculate the numbers as ratios of those not interested to those interested by country, and show our results in table 3.

It is obvious that the first irregularity is amongst Greek respondents, where half are not interested in taking part in the study! For the population without

Table 3. People not interested in the study to those interested

Language	BE	CZ	DE	GR	SK
Not interested	30 %	11 %	26 %	107 %	30 %

Table 4. Fraction of the respondents interested and eventually enrolling

Language	BE	CZ	DE	GR	SK
Interested	44 %	56 %	52 %	32 %	42 %

Table 5. Distribution of types of answers to the change of data usage

Language	BE	CZ		DE		GR	SK	
Sex	M	M	F	M	F	M	M	F
Declined	16 %	10 %	13 %	21 %	23 %	25 %	9 %	3 %
Same bid	39 %	47 %	42 %	51 %	45 %	50 %	49 %	40 %
Revised bid	45 %	42 %	44 %	28 %	32 %	25 %	42 %	57 %

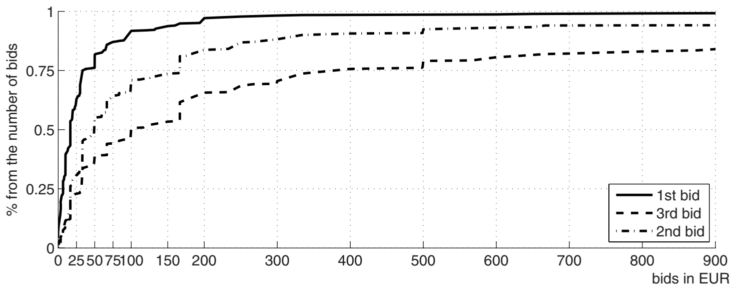


Fig. 1. Cumulative distribution of bids – people bidding in all three scenarios

early drop-outs, we get very similar statistics for those who did not fill their e-mail address (table 4). The numbers are fractions of *took_part* / were_interested.

The differences are not so large in this case but Greek participants are again the most cautious among all the nations.

An important part of our experiment was assessing people's sensitivity to commercial exploitation. Questionnaire 2 was requesting respondents to submit a bid compensating their participation in the experiment involving a commercial partner, and were also given a chance to opt-out. The following table (table 5) shows divisions of participants according to their answer for the 2nd bid.

Table 5 seems to present that participants from the two Central European countries (CZ and SK) were declining participation less often than respondents from other countries. (It might be an interesting fact related to recent history of controlled political environment requiring obedience from citizens.)

The first plot we present (fig. 1) shows bid distributions of people who entered values for all three possible uses of data. The x-axis depicts value of bids in EUR while the y-axis shows the fraction of bidders who entered bids of value lower or

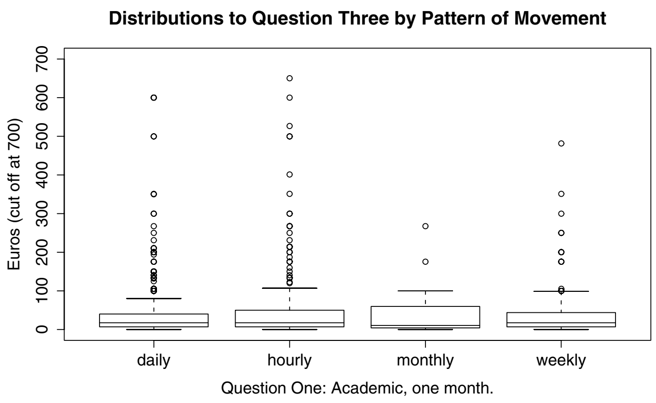


Fig. 2. Correlation between distributions of bid values and frequency of movements

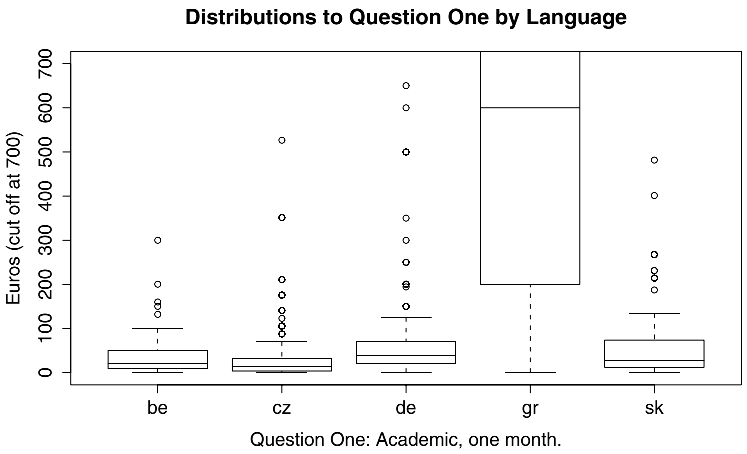


Fig. 3. Distributions of 1st bid values per country

equal to a given amount. One can see that median of the bids increased about 3 times when comparing the bids when the data were to be used only for academic purposes and for commercial purposes. The extension of the study period from one month to a year yielded only twofold increase in the bids (again measured in medians).

We expected the value of bids to be related to how much people are travelling or, in general, moving irregularly. Figure 2 depicts the distributions of bids according to regularity of movement, and refutes this assumption. The third quartile for hourly movements is higher that for daily or weekly movements, but monthly movements have an even larger third quartile.

We have already described several results demonstrating the increased sensitivity of Greeks to possible breaches of their privacy. Figure 3 gives an overview of bids by countries. You can again see that the Greek perception of privacy is much different from all other countries.

5 Conclusions

We hope to distill some more information from the experiment described above but we can already list the most interesting facts. There is about ten percents of people bidding below 1 EUR. We believe that this fact comes from curiosity and enthusiasm of participants. We received amount of feedback expressing interest in the results when the deception text (stating the improvement of the network quality as the main goal of the study) was sent out.

Surprisingly, we have not found any correlation between valuations and the way the respondents commute or move in geographic terms. The results do not show any such connection which is slightly contradicting the results of the Cambridge study [5].

The highlight of the experiment is the evidence of Greek sensitivity to possible privacy breaches. This almost certainly follows from an eavesdropping scandal [7]. Top politicians were being wiretapped for a period of eleven months during and after 2004 Olympic Games. This was confirmed at the beginning of February 2006 by the Greek government – just two months before our experiment actually took place.

The revised bids were 2.5 to 3 times higher (in average) than the first bids. We are of the opinion that this was the moment when participants realised that their data might be used for purposes hard to foresee. The third bids increased only twice from the second bids. This contradicts an assumption of proportionality between value of bids and amount of data collected.

Basic results confirm results of the Cambridge study – *e.g.* medians of bids are 20 GBP and 33 EUR for non-commercial use of data.

Acknowledgments

This project has been undertaken within the FIDIS Network of Excellence activities. George Danezis is supported by a grant from the Flemish Research Council (FWO). This proposal has been initiated through ongoing discussions with Jozef Vyskoč, and Claudia Diaz. Their assistance, together with the help of Vasiliki Andronikou, Lothar Frisch, Dana Hašanová, Stefan Köpsell, Eleni Kosta, Pavol Kosta, Pavol Kustar, Andrej Trnka, has been crucial for this project to succeed.

References

1. Syverson, P.: The paradoxical value of privacy. In: Second Workshop on the Economics of Information Security (2003)
2. Boucher, P., Shostack, A., Goldberg, I.: Freedom System 2.0 Architecture. Whitepaper, Zero-Knowledge Systems, Inc. (2000)

3. Law, G.: Anonymity Declines as Zero-Knowledge ends Web Service. PC World (2001)
4. Main, F.: Your phone records are for sale. The Chicago Sun-Times (2006)
5. Danezis, G., Lewis, S., Anderson, R.: How Much is Location Privacy Worth? In: Fourth Workshop on the Economics of Information Security (2005)
6. Hann, I.H., Hui, K.L., Lee, T.S., Png, I.P.L.: The Value of Online Information Privacy: Evidence from the USA and Singapore. In: International Conference on Information Systems (2002)
7. Danezis, G.: Government communication illegally wiretapped in Greece. EDRI-gram (2006), http://www.edri.org/edrigram/

The Value of Location Information
(Transcript of Discussion)

Vashek Matyas

Masaryk University, Brno

Bruno Crispo: Was the usual behaviour of the participants affected by the experiment, did they have to do anything specific?

Reply: No, they didn't have to do anything except if they are used to switching off their mobile for the night they were told that they have to have their mobile on all the time, that might be the case for a few of them, other than that, nothing.

Bruno Crispo: The conditions of presenting the experiment were exactly the same in all the different sites?

Reply: Yes.

Tuomas Aura: Did you ask how much students would pay in order to be allowed to participate?

Reply: No, we didn't ask that, and I believe that we would get 10% or 15% of people even giving us money to be able to participate.

Marios Andreou: Are you going to reveal to the participants at the end?

Reply: Oh yes, we wouldn't get clearing through the management or ethical committees if we didn't promise to reveal the nature of the experiment. This is a fair approach, used in many experiments in Social Sciences apparently, and I was told that there are even approaches that are considerably worse than this one.

Ross Anderson: Some of our psychologists suggested that it was a wicked thing to do to promise money and not pay any, because in psychology they're paying money to experimental subjects all the time.

Reply: We had that discussion. They accepted finally the logic that since the people are actually not selected for the experiment, they don't have to get the money. When we review the details about the true nature of the experiment, we could mail them to ask them, why did they take part. I believe that it might be a fruitful approach, definitely structuring the questions will be of a very tricky nature.

Frank Stajano: If you've already hacked them off by not giving them any money, and telling them, aha, they might not. [Laughter]

Tuomas Aura: It's like you're sent this marketing survey, and you're told that after filling this you might have the opportunity of winning some gifts.

Reply: No. Here, it would be like this kind of survey, but before you get that you would get the first form where they would ask, we want to send you this four page form, how much would you charge us. I believe that this is the nature of the experiment, not getting the four page form immediately in the first instance.

Tuomas Aura: But it's where their expectations might lead, especially those who ask you to pay 100 euros.

Reply: Oh no, they knew that it's an auction, and they knew that the budget was limited. Sorry, I didn't put this forward at the start. They knew that we will select only a subset of the parties who are interested, and we will pay them the bid that comes lowest from those who will not be taken in, so it was based on the normal auction principles. Many people put in money that would buy you a couple of beers, or it would basically reward you for the hassle of filling the form, and maybe not switching the mobile off for the night. When they were told that there might be a commercial use of the data, many of them obviously thought, OK, we will charge more. The data that we have shows that they are doubling the price basically. Why they are not putting ten-fold, I don't know. I am not an economist.

Ross Anderson: When we did the initial experiment we left a period of time, I think it was a few days or maybe a week, between first asking them whether they would participate in the experiment for research, and then asking them whether they would be prepared to put in a revised bid for commercial use. Did you leave a gap, or did you just ask them the next day?

Reply: No, there was no gap here, the questions were coming one after another. They could have split the answers, because they had the access password, and their email was used as a login, but I don't think many of them used this, I think that most of the people filled the answers immediately after another.

Tyler Moore: Well you might get a correlation there. With the commercial use, juxtaposed right next to it, they're going to automatically assume, this just has to be worth more otherwise they wouldn't be asking me this question.

Reply: There are both arguments against this and for this, I agree. As you will see the results show there is no difference.

Mike Bond: There may be a difference between how people would react to selling their location privacy to a party they have an existing relationship with, versus to a party they have a new relationship with. Maybe if they're already your students at your University, then you've got the existing relationship.

Reply: It was worded in such a way that the relationship here was of a nature something in-between, it was not entirely a new relationship, because this was a commercial partner of your operator that was in contract with the partner at the time when you filled in the data. So if you trusted your mobile phone operator to sign up to reasonable contracts only, then you would probably believe that this was acceptable commercial use. It was not just anyone coming for the data.

Mike Bond: It would just interest me how it would affect the price of bids if you smack a familiar logo on it.

Ross Anderson: I suspect that the significant price points might be whether the data can be resold.

Reply: Then it comes to the issue of not only who will use the data, but in what kind of usage, if it's a usage where you will do statistical analysis only, or where you can track individual people.

Ross Anderson: This is perhaps one of the things people have to explore in the economics of privacy, the extent to which you can make a complete contract. One of the problems with people anticipating privacy abuses of the data, is that they've no idea what the thing will be used for; the average person has no idea, for example, of the secondary uses made of their medical records, and no idea of the extent to which commercial data, things like loyalty card information, is traded between companies that do profiling. Now if you start getting into a world where all this stuff is for sale, then the unforeseeability, from the point of view of the individual who's trying to make a privacy contract, makes the contract more difficult, and either might result in people giving up, or it might result in the prices being high.

Tuomas Aura: But I think it might go the other way, that people stop caring, so in the future you might actually establish that a Tesco loyalty card gets them some cheaper insurance.

Ross Anderson: That's certainly the way it happens with the credit reference agencies.

Tuomas Aura: In the US, kids are told that when they are 18 you have to get a credit card and use it, because you establish a credit record, even if you don't need any credit.

Bruce Christianson: You don't need it now, but…

Tuomas Aura: I think mobile phones would be another possibility, the phone operator gave you free minutes, or free text messages, how many free text messages do they have to give you, for allowing them to check your location and use the data for whatever they want to use it.

Frank Stajano: But they can basically do that because there's so much fine print that you don't know if your contract doesn't already say that: we have to have your location because otherwise we can't route calls to your phone, and they can do what they like.

Ross Anderson: The privacy scare in America yesterday[1], was about the plans of Google, Microsoft, and others, to wire up American cities for free wifi access. The deal there is, that Mr Google will give you free wifi access with a low bit rate throughout San Francisco, you can give him twenty bucks a month, or whatever the subscription is, you get hi-speed wifi access, but presumably the small print says that if you take this free low-speed wifi access, he gets your location history.

Now we're about to see a very large-scale social experiment of precisely the subject matter of your paper, and I wouldn't be surprised if the people at Google start looking at your work once you've published.

Frank Stajano: Even if they give you £20 a month, or whatever, they're still getting locations-data right, so there's no difference to you.

Ross Anderson: Well the interesting thing is, what can Google get from providing free wifi access? Now if I use search all the time to go back to the Lab to get my emails, he's not going to get my content, but he can obviously get my location information.

Frank Stajano: But he would whether you're paying the £20 or not.

Ross Anderson: Well, so is there a market for somebody to provide software that makes your laptop into an address at Java device that changes your MAC address every six hours, or whatever. You could actually sell that for money.

Bruce Christianson: Something like Hedy Lamarr's frequency hopping[2], but with address hopping.

Michael Roe: There's a strange economic inversion here. If you're paying money for the service, then to identify your subscription, you've got to be known to the service provider. But if there's a scheme where anybody can get access for free, then you would better to pretend to be lots of pseudonyms. So it's exactly the wrong way round, by not paying for the service you get more privacy than if you pay for it.

[1] See `http://www.forbes.com/feeds/afx/2006/04/06/afx2653226.html`
[2] See `http://w2.eff.org/awards/pioneer/1997.php`

Ross Anderson: In that case the hardware project is to build a device that makes one laptop look like twenty, with constantly changing addresses, so that you get the high bandwidth access, and the privacy, at the same time.

Frank Stajano: Except if all twenty are moving at the same time then it is discovered, this cluster is your laptop.

Steven Murdoch: You don't need to be a MAC address in software. Only for the data packets the association uses the hardware MAC address.

Bruce Christianson: That would be the outcome, gangs of roving youths.

Steven Murdoch: Well all the protocols I've looked at for an association stage, you can't spoof the packets.

Matt Blaze: If Ross is coming back to a specific site with his spoof MAC address, the movement of the laptop user with unfamiliar MAC addresses tunnelling back to Ross' home is still going to reveal the same thing as his MAC address would.

Ross Anderson: Well there's a wonderful PhD project for someone to do a proper implementation with a peer-to-peer system of address-agile laptops in San Francisco with a privacy overlay, and all the other engineering that you'd need.

Reply: Yes, you have to keep warning the addresses not to use the constant share of twenty that you would use. Anyway, these are the possible approaches that we thought of how to separate out the group of people who would be in for the money.

Srijith Nair: Isn't it possible to just tell them, we have run out of funding, and we can't pay you anything, but can we still use your data? The people who respond yes will be people who are there for fun, not people who are in for the research projects.

Ross Anderson: But then you'd have to own up afterwards, and deceive them as well?

Reply: But it might be an option.

Peter Ryan: So how do you sleep at night?

Reply: A couple of beers is always a help. When we compare our data that we got to the basic findings of the Cambridge experiment, the valuations are roughly the same. If you compare the basic academic or research use, and the possible commercial use, they are doubling. There was an indication that travelling and movement for the Cambridge experiment was a sensitive issue due to the fact that (as I was kindly informed yesterday, having studied in Cambridge for four years in the past and not known about it) the students are not supposed to travel much out of the town, which is not the deal in countries where we undertook the experiment[3].

We would be definitely interested in suggestions, how to get the real perceived value, how to filter out those people who would be in just for the fun, or just being able to take part in this research. Any suggestions to the experimentary design are also welcome.

Tuomas Aura: Following Srijith's ides, you could now send a letter or an email, I suppose, to the participants, and say, well we've used our budget, but if you'd still like to take part without any payment, you can do that.

Reply: Yes, OK. That's even better, yes, good one.

Mike Bond: Then you can spot the people who are only in it for the money, because they won't take part in that second stage, so you could identify the people who are just beyond the highest bid.

Reply: If the people who respond say, no, then this would be the indication that these are the people who were in for the money, and these are the people who we would be interested in. If we do it along the lines that Tuomas just suggested, then this filters out the people who are in just for the fun, because these would say, yes, I will take part anyway.

[3] Precincts of the University and Residence, pp180-1 in chapter II of the Ordinances of the University of Cambridge (available on-line at `http://www.admin.cam.ac.uk/univ/so/pdfs`)

Update on PIN or Signature
(Transcript of Discussion)

Vashek Matyas

Masaryk University, Brno

We promised a year back[1] some data on the experiment that we ran with chip and PIN. If you recall, it was the first phase that we reported on here last year, where we used the University bookstore, and two PIN pads, one with very solid privacy shielding, the other one without any. We ran 17 people through the first one, 15 people through the second one, and we also had the students do, about half of them forging the signature, half of them signing their own signature, on the back of the card that is used for purchasing books, or whatever. We had a second phase of the experiment, after long negotiations, and very complicated logistics, with a supermarket in Brno where we were able to do anything that we wanted through the experiment for five hours on the floor, with only the supermarket manager, the head of security, and the camera operators knowing about the experiment. So the shop assistants, the ground floor security, everybody basically on the floor, did not know about the experiment. That was one of the reasons why the supermarket, or management, agreed to take part, they wanted to control their own internal security procedures.

We had to create our own accounts, and we had people using these accounts with real cards, but not compromising their own PINs. Comparing the results from the first phase and this second phase, the shielding really matters in the bookstore experiment, this was not confirmed in the second one. I believe that there are two reasons for that. In the first case we used really heavy security shielding, the shield around the keyboard was the most extreme case of shielding that I've ever seen. In the second case, in the second phase, the shielding was negligible, and it played basically no role, from the angle of the observers in the shop. What played a critical role, as you can see here, was basically the assertiveness, (or aggressiveness) of the bad guys, the observers. We had three groups of observers, two of them scored about a quarter of the PIN digits at the till, and as you will see, those performing really well were able to observe correctly about two thirds of the digits at the PINs. I believe that these numbers are more indicative, these are the percentage of the correct digits that the observers would get from you in a shop if they watch you typing in your PIN.

In the first experiment the percentage was slightly higher because the bookstore was a closed environment, where we did not have more customers coming in, it was just one customer, and the people were really able to focus, and get around that customer. In the supermarket experiment, it was Friday late morn-

[1] D. Cvrcek, J. Krhovjak, V. Matyas, *PIN (and chip) or Signature: Beating the Cheating?*, LNCS 4631, pp 69–81.

ing and early afternoon operations, so you had many customers in the shop, it was not always possible to get the observers right in front, and right behind the guy who was shopping. In both cases their primary task was to observe, their secondary task was not to be spotted by the subjects.

George Danezis: The question is, what could someone do if they spot one of your observers. I as a customer, I could maybe hide my PIN when I type it, but fundamentally you can't really tell the shop assistant or the security guard, oh you know what, he's been looking at my PIN.

Reply: Why wouldn't you say, tell it to the security guard?

George Danezis: I don't know, how many people here would?

Mike Bond: Security guards tend not to wear hats in the UK, so you need to find a security guard.

Reply: These were uniformed security guys who stood behind the tills, definitely less than 15 metres away, so you could just call, hey, come and help me, this guy's watching me. So yes I believe most of the people there would report.

George Danezis: Did anyone report?

Reply: No. At the end of the four hour period, one of the till assistants started watching the guys because she saw the same faces running round for four hours. [Laughter] She has a phone connection to security on the first floor of the supermarket management offices, but she didn't report anything; our guys just told us that she looked a little bit more cautious.

Tuomas Aura: That's funny because a long time ago I worked in a supermarket, and I think that normally people working in the store are thinking, oh, no, something's wrong, we've got to catch them.

Reply: Yes, but first they have to get the suspicion, and the point is how long it takes them to get there, if it takes them four hours with the same people running on the floor. We did not have the same people running around the same till all the time, but we had a row of a dozen tills, where we had three groups of participants. These groups didn't change formation so it was the first group, second group, and third group, who were just moving and going between different tills. I would think that people would figure out that there are people who are always rushing there to be one of them in front of the customer, one of them being after the customer; they didn't.

I will get to the signatures, that's the last slide. In the first case we used for signature verification a guy who owns (and performs signature checks in) a jewellery shop, so the success rate there was only 30%, he was quite thorough in his checks, as I mentioned last year. In this followup case every one went through at the first try. I've seen some of the signatures, some were pretty much plausible, but some of the signatures, with a reasonable way of thinking you would never, ever accept such a signature comparing it to the card. In some cases the assistants didn't even bother looking at the signature itself, they were just happy with the people signing the paper, and giving them the paper that's signed, so all the people went through at the first try, no-one had to sign twice. And my personal experience is that in the Czech Republic they sometimes can get a bit thorough so that you have to sign more than once because your signature doesn't look entirely the same. Because my shopping usually goes by two orders higher than this small shopping that we did in the supermarket, I asked in the supermarket whether they have some thresholds over which the thoroughness of the check of the signature should be better; the response was "no", that it's up to the individual decision of the shop assistant, and they should check everything.

Matt Blaze: What is the liability if there's a fraudulent transaction, is it the store, or the credit card?

Reply: I don't know about this specific supermarket, but generally they can have liability which is limited, typically it's 150, 160 euros.

Matt Blaze: So it may just not be worth it to risk offending customers by checking at all? If you think it doesn't match, then you've got a dilemma if you're a shopkeeper, because if you ask you may offend the customer, and risk losing the sale, right, why do you think I'm a thief?

Reply: Well I definitely agree with that, the point is that some of the assistants didn't bother checking the signatures.

Ross Anderson: One of the reasons often cited is that the shop assistants were reluctant to say, this card is a forgery, even when it was blatantly in an experimental context. Now I can recall some other work, that you got shop assistants challenging people doing credit card forgery more or less only where the rewards offered by the credit card brands were actually passed on to the shop staff. If they were just trousered by the shop, then the shop staff wouldn't bother. So another thing to look at here is the policy of the supermarkets in terms of rewarding shop staff detecting stolen cards, if a stolen card is worth $50, say in America, does the shop assistant get $50 or $10 or nothing?

Matt Johnson: With the new self-service tills they're doing in some of the supermarkets over here, they don't do chip and PIN when you pay with your card, if it's above £100 then someone comes over and gets you to sign a piece of paper, and if it's below that it just goes through without any checks whatsoever.

Reply: Basically you enter the PIN, and you are still asked to sign that paper. I personally tested this on the day when we went to the experiment just to check the floor and everything. I falsified a signature that I hadn't seen before, I just knew Marek's name, I didn't know whether he signs his first name in full or just the initial, I obviously did the contrary, the assistant looked at the signature, it was considerably different in my handwriting, and it was not saying M Kumpost, but he was saying Marek Kumpost, she looked at that and she said, oh, but the PIN was alright, go ahead, that's fine.

Anyway, the second phase indicates to us that PINs are slightly better than signatures, but as you see, the figures for the assertive guys getting two thirds of the digits of customers' PIN is quite a good success rate. The secondary observation of no reaction from the shop ground security or assistants, was quite interesting, because it was a five hour experiment.

Innovations for Grid Security from Trusted Computing

Protocol Solutions to Sharing of Security Resource

Wenbo Mao[1], Andrew Martin[2], Hai Jin[3], and Huanguo Zhang[4]

[1] Hewlett-Packard Laboratories, China, 112 Jian Guo Road, Beijing 100022, China
[2] Oxford University Software Engineering Centre, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK
[3] Huazhong University of Science and Technology, Wuhan 430074, China
[4] Wuhan University, Wuhan 430072, China

Abstract. A central problem for Grid (or web) services is how to gain confidence that a remote principal (user or system) will behave as expected. In Grid security practice at present, issues of confidentiality and data integrity rely on weak social trust mechanisms of “reputation maintenance”: a principal who is introduced by a reputable party should hopefully behave in “best effort” to maintain the reputation of the introducer. As will be discussed in this paper, this gentleman’s notion of trust is insufficient for a large class of problems in Grid services.

The emerging Trusted Computing (TC) technologies offer great potential to improve this situation. The TC initiative developed by the Trusted Computing Group (TCG) takes a distributed-system-wide approach to the provisions of integrity protection for systems, resources and services. Trust established from TC is much stronger than that described above: it is about conformed behaviors of a principal such that the principal is prohibited from acting against the granted interests of other principals it serves.

We consider that this stronger notion of trust from TC naturally suits the security requirements for Grid services or science collaborations. We identify and discuss in this paper a number of innovations that the TC technologies could offer for improving Grid security.

Keywords: Trusted Computing (TC), Trusted Computing Group (TCG), Grid Computing, Grid Security, Behavior Conformation, Remote Platform Attestation, Secure Multi-party Computation, Secure Virtualization.

1 Introduction

A computational Grid [15,18,20] is a distributed computing system comprising a number — possibly large — of physically separated resources, each subject to their own various security, management and usage policies. It is intended to support a variety of users who may be working on a number of common tasks and have similar resource requirements. A system of such collaborators and resource providers may be described as forming a virtual organization (VO). Some such VOs may be very dynamic, called into being for a single, short-lived task. In the most general setting, a VO of users and resource providers is geographically distributed and in different trust and management domains. These domains can span governmental, industrial and academic organizations. This implies, even demands, that strong security mechanisms be in place so that the Grid services can be used in a secure and accountable manner.

Therefore, two essential characteristics of Grid security are:

System Behavior Conformation. Because typical Grid resources — infrastructure, applications, instrument or data — have critically high importance and value, a Grid security strategy should be based mainly on attack prevention. While entity authentication is an important means for controlling access to resources and can also achieve attacker identification after an attack, it does not provide an effective means of attack prevention. This is better achieved with a behavior conformation mechanism: an entity and its supporting computing system is attested that they have a restricted (and desirable) behavior which cannot (easily) lead to any serious damage.

Group-Oriented Security. Resource sharing in a Grid VO is, by definition, a group-oriented activity; a grid security solution must support such capabilities. Many accounts of Grid design describe use scenarios entailing research data being shared by a group of scientists, large scientific instruments which must be operated by a group of users at the same time, or *ad hoc* collaborations such as a conference discussion among a group of entities (who therefore need to be served with a shared conference key). A useful (and difficult) case of group-oriented security is in the form of *secure multi-party computation* (SMPC) where proprietary data are input to a VO's common computational task in such a manner that no member of the VO should gain access to data input by any other participant after the joint computation.

Several aspects of Grid security are well-explored: the use of public key cryptography, with PKI identity and attribute certificates is quite well explored (and ongoing) for assuring identity of users, servers, and potentially software itself. These may be supported by a range of policy decision tools to enable authorization mechanisms. Most grid applications entail code written in one place being executed in another. The problem of potentially malicious code and a trusted host is met by techniques such as sandboxing, code signing, or virus checking, or simply through strong accounting so that if the code's execution causes substantial cost, its owner is required to pay substantial sums.

The dual of the last problem — trusted code required to run on a potentially malicious host — is harder to address. The possession of a host identity certificate is no guarantee that its administrators are not interfering with the execution of software, observing its inputs and outputs, or simply not offering the promised quality of service. Techniques of code obfuscation may make reverse engineering of software arbitrarily hard but for practical purposes it is unsafe to distribute code and assume that no one will be able to break or subvert it. Theoretical approaches from cryptography and/or statistics hold promise, but are hard to integrate with existing code, or require substantial overheads in order to work.

In recent years, increased computer security has been the goal of many efforts made by the computing industry. Among the many ideas, we are specifically focusing on the Trusted Computing (TC) initiative by the industrial standard body, the Trusted Computing Group [29]. The purpose of the TCG is to develop, define, and promote open, vendor-neutral specifications for trusted computing. It begins with a simple idea: integrating to a platform a low-cost tamper-resistant hardware module to enable and manage data and digital identities more securely within the platform's environment, protecting them from external software attack and physical theft. The TCG work has so far been developed with sufficient innovations to achieve its goal. These include hardware building block and software interface specifications across multiple platforms and operating systems' environments. The TCG's open specifications (versions 1.1b and 1.2, available at the "Downloads area" of [29]) not only define reasonable notions of trust and security, but also provide concrete mechanisms to achieve protections by means of policy and trusted environment conformance.

Many authors have remarked on the suitability of these systems for distributed computing or even grid computing but the details are sketchy. Recently, as the TCG technology — hardware modules and the related device drivers — is becoming available, it is timely to consider how it may in practice assist in some grid application scenarios. We observe that the TCG mechanisms for policy and trusted environment conformation can provide a needed role in Grid security. This is particularly suitable for our two Grid security characteristics listed above. In this paper we propose an innovative approach to Grid security from Trusted Computing effort.

The remainder of this paper is organized as follows. In §2 we consider Grid security requirements, illuminating these by some use cases in §3. In §4 we overview the current Grid security solutions and identify their inadequacy with respect to our two characteristics for Grid security. In §5 we overview the Trusted Computing technology. In §6 we consider Trusted Computing technology as the complementary solution to the identified problems in the Grid security. Finally in §7 we provide discussions on issues of the TC implementation and deployment.

2 Grid Security Requirements

The US Department of Energy (DoE) Office of Advanced Scientific Computing Research published a report which provides a good summary of the requirements for Grid security [15]. The Grid requires a security infrastructure with the following properties:

I) Ease of use by users.
II) Conformation with the VO security needs while at the same time working well with site policies of each resource provider site.
III) Provisions for appropriate authentication and encryption of all interactions.

In the sequel, we shall refer to this set as the "DoE Grid Security Requirements." We hold the view that DoE Grid Security Requirements II and III are compatible with our two characteristics for Grid security. More clarifications will be provided in the remainder of this paper.

3 Use Cases

By way of illustration, we record some realistic security requirements of some Grid applications.

3.1 climate*prediction*.net Clients

The climate*prediction*.net project seeks to use the best available climate models to produce large *ensemble-based* forecasts of the development of the world climate over the next 50 years. The approach taken is to distribute this model to tens of thousands of participants across the globe, inviting them to run it for a period of about six weeks, returning the results when complete to one of a number of project upload servers. This ensemble of results forms a *Monte Carlo* simulation, allowing climate scientists to construct a probabalistic model of the likely scenarios for climate change [31].

This approach to distributed computing was made popular by the `SETI@home` project [30]. There, data from radio telescopes is divided up and distributed to thousands of participants' computers, each of which analyzes that data for signal patterns which may indicate extra-terrestrial intelligent life. Many other projects have adopted a similar approach: Einstein@home (analyzing data to search for gravitational signals from pulsars), Folding@home (modelling protein behaviour to understand a variety of diseases), distributed.net (amongst other things, achieving brute-force cracks against cryptographic algorithms), and so on.

The projects differ in the nature and scale of the task — climate*prediction*.net being *simulation*; `SETI@home` being *search*; work units in the latter complete in a matter of hours rather than weeks for the former, and their file transfer requirements are measured in kilobytes rather than megabytes.

The projects clearly have much in common, and indeed, many now implemented using the BOINC [2] platform. This is a general-purpose open-source platform into which application scientists can install particular simulations or search software, experiment parameters and initial data, and from which a complete community project of this kind can be run.

All of these projects must address the same central problem: it is quite feasible for participants to return data which appears to come from running the downloaded software, but in fact arises from a different source. Either the participant has modified it (and some `SETI@home` participants have done this, seeking to help the project or to boost their personal ratings for work units completed) or they have completely fabricated the results (for a host of possible reasons). climate*prediction*.net has implemented a hashing checksum to guard against casual tampering or creation of results — but has declined to enter an unwinnable "arms race" against the determined hacker [13].

In both projects, the impact of such behavior might be substantial: false positives or negatives in the search for anomalous signals in `SETI@home`'s case; biased statistics in the climate*prediction*.net case. The first project has been very successful in attracting participants, and so frequently has far more compute power available than it has data to process. As a result, searches can be duplicated several times over among participants, and non-matching results simply discarded. The second project, with its much greater resource requirements and commitment, has a lower number of participants, and cannot afford to duplicate model runs on a large scale. Happily, though, because the entire modelling effort is a statistical one, individual out-lying results are not a significant problem, provided no systematic bias is introduced. For climate*prediction*.net, then, it suffices to duplicate a small random sample of runs, and to use a pairwise comparison to estimate the accuracy of the whole ensemble.

For both projects, a much more robust solution would be to gain some kind of assurance that the software running was the intended software, with the intended inputs, and that the results returned are those created by the software, unaltered.

3.2 Grid Data/Compute Nodes

Clearly, any kind of subversion which may arise in the climate*prediction*.net host could equally well arise in a more tightly-coupled grid node also. With the system administrator's connivance or otherwise, the host might appear correctly to process grid computing jobs, but might falsify results, or might retain a record of "interesting" inputs or outputs.

Many instances of such concerns are already known:

climate*prediction*.net Upload Servers. The climate*prediction*.net upload servers are themselves donated resources from sympathetic academic departments across the world. Because the whole dataset is massive and cannot easily be collected in once place, it is stored, and must be processed, at these donated nodes. However, no genuine guarantee can be had of their ability or willingness to enforce data integrity (storing results without perturbation), or to compute derived results accurately.

Bioinformatics and Databases. A similar concern arises in the bioinformatics arena, where much work relies upon queries against common databases. Information about what those queries are is commercially sensitive — it will indicate an area of study to a competitor — and so although those databases are public they will often be copied and held (at substantial cost) in-house, because their system administrators are not trusted to refrain from harvesting query information. An alternative approach is to run 100 queries at a time, only one of which is a genuine request, in order to confound anyone looking for patterns. Neither is a very efficient scheme.

Sensitive Code. A further deployment scenario arises in the Integrative Biology project, where the security of data is not a major concern, but the code being run represents the fruit of much research effort — and so is valuable intellectual property. The heart modeller whose career has been devoted to making a particular model may wish to run it on a high-performance computer but does not want to run the risk of the administrator (or, worse still perhaps, another user) of that machine taking a copy of the code.

In each case, one would wish either to certify the whole software stack ahead of time, or to subject it to audit, and to then have a remotely-checkable guarantee that the audited software is that still being run. We would wish to do this in a platform-independent manner, since in an ideal grid context we should neither know nor care which particular host is running our software or hosting our data.

3.3 Medical Informatics

Increasing complexity of electronic patient records raises a requirement for forms of mandatory access control. Different parts of each patient's records should be accessible by varying sets of people: administrative staff, nurses, general practitioners, specialists. For purposes of epidemiology and other research, anonymized or pseudonymized records should be available. Although attempts are being made to keep the access rules simple, the current societal interpretation of issues of consent mean that some detailed fine-grained rules are required. If an element of a patient record is stored in an encrypted form, the decrypted form should only be available to individuals in selected roles, and where suitable facilities for record-keeping and audit exist.

If follows that a reasonable requirement might be for a clinician receiving sensitive information to be prevented from passing it to a third party. At the moment, no strong mechanisms exist to enforce this. As a result, the UK NHS, for example, uses separated networks, which drives up costs, and can nevertheless not give strong guarantees of separation (because the network is so large that we cannot reasonably imagine it is homogeneous or completely audited).

Such issues of *end-to-end security* [12] are not at all addressed by present Grid solutions; nor is it clear how they might.

4 Current Grid Security Solutions

4.1 Authentication

The Grid Security Infrastructure (GSI) [19] and MyProxy [24] are two important elements of many current Grid security solutions.

The GSI, which is the security kernel of the Globus Toolkit [22], provides a set of security protocols for achieving mutual entity authentication between a user (actually a user's proxy which is a client-side computing platform) and resource providers. Entity authentication in the GSI protocols involves straightforward applications of the standard SSL Authentication Protocol (SAP) suite [21]. These standard applications can be considered as a "plug-and-play security solution." They achieve quick deployment and ease of use. As a result, the Grid security protocols in the GSI are two-party mutual authentication techniques.

Each party has a public-key based cryptographic credential in the formulation of a certificate under the standard public-key authentication infrastructure PKI X.509 [23]. The use of the standard PKI in Grid security is not only suitable for the VO environment, but also has an important advantage: single sign-on (SSO). The latter means that each user only needs to maintain one cryptographic credential. As always, any security solution must not demand the user to invoke sophisticated operations or tools.

Using PKI requires each user to hold a private key as their cryptographic credential. This can be a demanding requirement for many users without a secure computing platform in their locality. MyProxy provides a lightweight solution. It uses an online credential repository which can deliver temporary Grid credentials to the end user. This is achieved via simple user authentication mechanisms such as password. This can be enhanced via a one-time password such as through a SecureID card.

The combination of the GSI and MyProxy provides a credible solution to the DoE Grid Security Requirement I. The two-party authentication protocols of the GSI, however, do not provide an adequate solution to group oriented Grid security applications. For example, consider the DoE Grid Security Requirement III: the GSI cannot easily achieve a common key for a VO-wide encrypted communication.

4.2 Authorization

The Grid authorization landscape is far more varied. Products such as Akenti [28], Community Authorization Service [25], VOMS [1] and PERMIS [5] take a variety of approaches. Most make further use of X.509 certificates for identity or other attributes. Typically, it is up to a virtual organization to construct an authorization regime which enables it to meet the security requirements and policy of resource providers. These services are related to DoE Security Requirement II.

4.3 Secured Communications

For a host of reasons, it is seen as desirable to achieve integrity or confidentiality of data and control communications in Grid contexts. Although some have proposed using Virtual Private Networks for such a purpose, others have argued [10] that this is inappropriate. More commonly, transport level security (TLS/SSL) is employed. This has the benefit of being ubiquitous and highly interoperable, and supported by readily available hardware accelerators, but is emphatically a point-to-point solution.

Web Services Security [14] is potentially much more flexible, and in principle more efficient (since only selected elements of the communication are encrypted) — though present implementations do not realize this. WS-S takes a message level security approach by performing encryption at the Web Services layer, such as the XML messages. These solutions also make use of X.509 PKI. Observe that the services these latter solutions provide are orthogonal to DoE Grid Security Requirements.

Given the above, we can call the current Grid security solutions "plug-and-play PKI" for a conventional client-server environment. It is clear that two-party protocols based Grid security solutions neither directly nor effectively support a group-oriented security. Additionally, they do not have a inherent means for realising behavior control for a remote user and its client system environment. For example, WS-Security can achieve message encryption between a resource provider and a user. However, there is no way for a stakeholder in the resource provider to know whether or not the remote client environment is compromised (perhaps by a malicious code) even though it knows that such a compromise is equivalent to the nullification of the channel encryption service.

5 Trusted Computing

In 1999 five companies — Compaq, HP, IBM, Intel and Microsoft — founded the Trusted Computing Platform Alliance (TCPA). In 2003 the TCPA achieved a membership of 190+ companies, when it was incorporated to the Trusted Computing Group (TCG) [29] as a vendor-neutral and not-for-profit organization for promoting industrial standards for Trusted Computing technologies. The TCG takes a distributed, system-wide approach to the establishment of trust and security. It defines a concrete concept of Trusted Computing (TC). We may consider TC as the desired and conformable system behavior which is not only established and maintained in a platform environment, but can also be attested to a remote challenger.

The following four notions are at the core of the TC technology:

Trusted Platform Module (TPM): The TPM is a tamper-resistant hardware module uniquely integrated to a platform for conformed operation and secure storage. It is designed to perform computations which cannot be subverted by the platform owner, including the system administrator. These computations include some public key cryptographic operations (decryption and digital signature generation using a private key in the TPM), platform system status measurement, and secure storage. Each platform has a TPM.

Core Root of Trust for Measurement (CRTM): At platform boot time, the TPM measures the system's data integrity status. The measurement starts from the integrity of BIOS, then that of OS and finally to applications. With CRTM, it is possible to establish a desired platform environment by loading only well behaved systems. This is a strong requirement which is called "secure boot." The TCG also permits a slightly weaker measured boot which is called "authenticated boot." In the latter the TPM will permit loading of code which does not pass the measurement but will only securely record the status of that which has passed the measurement for attestation purpose (see below).

Root of Trust for Storage: The measured integrity of an executable is represented by a cryptographic checksum of the executable. This is then securely stored in a TPM. The TPM component called *Platform Configuration Register* (PCR) holds this data in an accumulative formulation. The TPM has a number of PCRs; each of them can be used to accumulate system integrity data for one category of system executables, *e.g.*, one PCR for OS's (a platform can run many copies of OS's, see §6.5) and one PCR for a family of specific applications. The stored platform environment status is maintained until system reboot.

Remote Platform Attestation: By using cryptographic challenge-response mechanisms, a remote entity can evaluate whether a platform's system has desired and conformed behavior. Remote platform attestation is the most significant and the most innovation element in the TC technology. With this capability, a remote stakeholder can be assured, with confidence, of the desired and conformed behavior of a platform. In §6.4 we will provide a concrete protocol specification to manifest the functionality of platform attestation.

We notice that with a platform having the above behavior, the TC technology has met resistances by being interpreted as providing for monopoly control over the use of software; trusted computing has its detractors [3,4]. The TCG considers this a misinterpretation because a TCG platform should be able to execute any software in the "authenticated boot" condition (see CRTM above).

Others argue [11] that market forces, combined perhaps with light-touch regulation and scrutiny, will help to keep the world sane. We may also observe that faulty software abounds and will help to keep the market from becoming completely controlled by any single party.

At any rate, we are able to avoid this controversial issue here. In the attempted TC application to Grid security there should be much less disagreement since Grid computing either requires behavioral compliance from an individual user as a condition for using remote resources, or implies federation and cooperation among a group of users.

6 Trusted Computing for Grid Security

We believe that TC technology can offer good solutions to Grid security problems for which current Grid security solutions do not play a role. Specifically, we argue that TC technology addresses particularly well the DoE Grid Security Requirements II and III.

6.1 Secure Storage of Cryptographic Credential

Unattended user authentication is an important feature in the Grid. This means that a user working in a VO is mainly doing so via their proxy. Work within a VO may involve dynamic sessions of resource allocation and hence require user entity authentication without having the user present. In the GSI, and in MyProxy, this is achieved by having a user client platform be issued a proxy certificate. The cryptographic credential of this certificate (*i.e.*, the private key matching the public key in this certificate) is simply stored in the file system of the platform protected under the access control of the operating system. In this way, the client platform does not need to prompt the user for cryptographic operations. The obvious danger of leaving a private key in the file space is mitigated by stipulating a short lifetime for the proxy certificate. The default lifetime of a proxy certificate in the GSI is 12 hours. Upon expiration, a new proxy certificate must be re-issued. We feel this is an unacceptable security exposure.

With a TCP containing a tamper-resistant TPM, it is natural to store a user's cryptographic credentials in the TPM, or under an encryption chain controlled by the TPM. In TC, each user of a platform can generate many copies of private keys with their matching public keys being certified in the standard X.509 PKI. A TPM can be configured to hold keys in a "non-migration" mode which will never reveal any private key (up to the tamper-resistance level for which a TPM is designed). Keys can also be configured as "migratabe," wherein key material can be explored with the owner's consent. Thus, even if a platform is under the control of an attacker, the attacker, though in this situation may be able to misuse the user's credential (still in a conformable manner), cannot retrieve any information stored in the TPM. Thus, in a TC enhanced Grid security setting, the protection of user secret key credentials can be substantially improved.

6.2 Sharing of Security Resource by Roaming Professionals

In GSI, MyProxy provides a lightweight solution to roaming professionals to obtain Grid services ubiquitously [24]. It uses an online credential repository which can deliver temporary Grid credentials to the end user. This is achieved via simple user authentication mechanisms such as password. A user shares a password with MyProxy server. Whenever and wherever the user requests for a cryptographic credential by authenticating to the MyProxy server, the server will generate a proxy certificate for the user and this includes the private key. The certificate is sent to the user, with the private key encrypted using the shared password. As we discussed in the previous section, a proxy certificate with a password encrypted private key form a weak security mechanism. GSI prioritizes ubiquitous services over strong security.

We should notice the most basic behavior conformation property of the TPM: prohibition of even the owner of the TPM from accessing certain protected data. Let a TPM have a public key for use by remote users, such that the decryption must only be possible inside the TPM and the result is not easily accessible even by the TPM owner, for example, the decryption result only exists in a memory location which prohibits the platform owner to access.

Now, a user who is not TPM equipped, perhaps because of a roaming professional whose home-base machine is a desktop, can use other people's TPM resource while obtaining a proper protection of her/his privacy, even from the TPM owner. Such a user may still use a MyProxy server to generate a proxy certificate (needn't be a short-lived one). The MyProxy server should encrypt the certificate using a public key of a given TPM, and make the certificate usable only by the user who should input to the TPM the correct password (also encrypted using the public key). The owner of the TPM equipped platform, if trying to gain an access to the user's proxy, must at least attack the password which the user has used in the protection of the certificate.

In this way, TC's conformed behavior property enables a secure sharing of security resource (the TPM). We notice that, although TPM will not become everywhere available overnight, use of the TPM as a shared resource (can even be remotely shared) in some applications, such as Grid security, can indeed happen within a short period of time.

6.3 Distributed Firewall for a VO

In a conventional organization a firewall plays an effective role in protecting the information assets of the organization. A conventional firewall relies for its function upon the notions of restricted topology and controlled entry points. More precisely, a firewall relies on the assumption that every entity on one side of the entry point (the firewall) is to be trusted, and any entity on the other side is, at least potentially, an enemy. Because many attacks are achieved via malicious connections which can be shielded by a firewall, firewalls are a powerful protective mechanism.

A Grid VO is typically composed of multiple physically distinct entities which are in different organizations who usually do not (entirely) trust each other. There is no longer a notion of a restricted network topology. The current Grid security solution does not utilize the notion of firewall based protection. A user (its proxy) enters a VO without bringing in its own computational resource. Such a VO is in a primitive stage: a user only uses resource "out there," rather than also contributing their own resource as well. In fact, many Grids have value precisely because every participant becomes a taker as well as a giver. Imagine the augmented value of a medical research collaboration which combines small databases of some limited clinical trials information scattered in various hospitals into global database available for access and search.

Bellovin proposed a notion of distributed firewall [16] which exactly suits the situation of a Grid VO. In a distributed firewall, a packet is deemed to be accepted or rejected according to whether it has an acceptable digital signature. The packet's acceptance not only depends on the validity of a signature, but also on the rights granted to the certificate.

At first glance it seems that the current Grid security solutions can already achieve a distributed firewall for a VO since these solutions also use public key cryptography and PKI authentication framework which enable the use of digital signatures. The main problem is that the short lifetime of a proxy certificate of any participant makes the packet-level signature verification a performance burden. We repeat that the acceptance of a signature in a distributed firewall application is not only on the validity of the signature in the conventional sense, it should also be judged on the firewall policy granted to a certificate. The short-lived proxy certificates used in the current Grid solutions are mainly limited to "identity certificates": these certificates are not suitable for distributed firewall use which needs refined policies associated to an IP configuration. We can call a certificate for a distributed firewall use a "property certificate."

With TC technology making multiple long-term (node and property) certificates available to each a platform, a Grid VO can readily implement a distributed firewall technique.

6.4 Attestation of Behavior Conformation in a Remote System

A Grid stakeholder has legitimate reasons to worry about whether a participating subsystem in a VO conforms to the VO's security policy. For example, consider the need for a remote platform, which is sending in a GridFTP query for some sensitive information, does indeed run the correct version of the GridFTP which will flush the downloaded data from the local memory without saving a local copy in the file system after using the data (or only save an encrypted copy). Likewise, a participating client in a secure multi-party computation (SMPC) task may also have similar concern with respect to its proprietary data input to a VO. In an SMPC, data input to a distributed algorithm (protocol) from each of the participating parties should be confidential to the group in such a manner that the group can jointly compute a result while none of the participant can gain any knowledge about input data from any other participants.

TC's notion of remote platform attestation is a ready solution for this sort of Grid services. Now let us describe how platform attestation can convince a remote user conformed behavior of the platform.

A TPM contains a number of registers called Platform Configuration Registers (PCRs). Each PCR accumulates cryptographic hash checksums of secure applications (software systems) which are currently running on the local platform. Let SA denote a secure application, *e.g.*, part of a protocol for GridFTP or SMPC, and let $H(SA)$ be the hash checksum of SA. Suppose that a remote user Alice initiates the protocol which causes SA to run on a TPM equipped platform (which we denote by TPM-Platform). Since the application is a secure one, Alice concerns whether or not TPM-Platform does run the bona-fide copy of SA. Protocol 1 (in the box) specifies a typical case of platform attestation to allow TPM-Platform to attest to Alice regarding her concern.

Protocol 1: Remote Platform Attestation

1. (In response to Alice's initiation) SA in TPM-Platform generates a public/private key pair SA_{pub}, SA_{pri} and sends them to TPM;
2. TPM creates $H(SA)$ and accumulates it into a PCR in the following formula

   $$PCR' \leftarrow PCR \oplus H(SA);$$

   TPM applies an "Attestation ID Key" (AIK) to certify (*i.e.*, digitally sign) the information about SA; we denote by

   $$Cert_{\mathrm{SA\_Start}} = \mathrm{Sig}_{\mathrm{AIK}}(SA, H(SA), SA_{pub}, PCR', PCR, Ctr);$$

   here Ctr is TPM's counter value which increases monotonously in each instance of authenticated boot and cannot be reset (not even by the owner of TPM-Platform);

3. SA sends $Cert_{\text{SA\_Start}}$ to Alice; she verifies the validity of the certificate using (the public) AIK of TPM; it is Alice's responsibility to deem whether or not to accept SA, *e.g.*, by checking if $H(SA)$ is the correct value (which should have already been publicized by another authentication server regarding SA); Alice shall also send a random challenge to SA in TPM-Platform;
4. SA responds by signing the challenge value using SA_{pri};
5. Upon Alice's acceptance of the response, she can be convinced that TPM-Platform does indeed run the bona-fide copy of SA in an authenticated boot session which is identified by Ctr;
6. Upon termination of SA, Alice can ask TPM to issue

$$Cert_{\text{SA\_End}} = \text{Sig}_{\text{AIK}}(SA, H(SA), Ctr);$$

having seen Ctr in $Cert_{\text{SA\_End}}$ unchanged from that in $Cert_{\text{SA\_Start}}$, Alice can further be convinced that the authenticated boot session of TPM-Platform has been maintained during the whole execution of SA; this assures Alice that SA has been running and then properly terminated in the correct (trusted) session in TPM-Platform.

The TC innovation in remote platform attestation provides a powerful solution to the integrity protection of resources. Integrity protection of resources is a serious problem which the current Grid security techniques cannot solve.

6.5 Securely Virtualized OS's and Services as "Vaults"

Using the notion of a virtual machine (VM) [7], an area of memory in a computing system can be isolated from the rest of the system to provide a simulated computer as if it were a separate computer. One piece of hardware can even enable multiple general-purpose OS's. Relations between these OS's can be configured to satisfy various access control policies. Moreover, on a TPM-platform, an access control policy for a VM as an object of other software systems (maybe other VMs) on the same platform can be conformed and attested to a remote user of the VM by applying Protocol 1 in the preceding section. Let's use "attested VM" to name a VM which has attested to a remote user an access control policy the user desires. Garfunkel *et al.* [6] consider that an attested VM can be a "lock down" OS (which they name "closed-box VM"). Such a lock down OS may only permit to run a given list of secure applications. Again, these secure applications can also have behavior conformation features which can be attested to a remote user. Thus, a lock down OS can serve a remote user a "vault" like service over a foreign platform. The "vault" on the platform is not even accessible by the platform's owner. The remote user (Alice) can send her data encrypted by a public key of the "vault" to input to an secure application running on the "vault" and then obtain the computation result which sent back to her encrypted under her public key. This achieves a secure guest computation, and on top of it SMPC (see §6.4) is practical.

Secure guest computation is very relevant to Grid services. In many enterprise organizations it is typical that many PCs run continuously while not being used for extensive periods of time, *e.g.*, outside working hours. Also, in many organizations typical uses of a PC involve word-processing like jobs which require minimal resource utilization by the prime PC user. According to studies by Microsoft [17] typical PC utilization is between 10 and 20 percent. A similar situation also applies to the servers environment, *e.g.*, [27]. With secure guest computation, it is realistic to suppose that large chunks of underutilized platform resources (enterprise PCs and servers) can be organized to provide services for external users (or applications). It is obvious that a stringent security policy conformation is necessary. A "vault" service can achieve exactly the needed stringency to protect the interest of the external users. For example, when faulty code used by a prime PC user crashes or hangs, the rest of the system services should continue serving uninterrupted.

6.6 Group-Oriented Security

Combining the distributed firewall technique of §6.3 with the remote platform attestation technique in 6.4, we can imagine a realization of a group-oriented security for a VO. As in the case of a physical group, in a VO there also needs to be an entity acting as the group manager or a stakeholder. The group manager is responsible for defining and managing the group security policies. These policies can be tailored to the setup of each site. The group security policy definition, setting up and management can be achieved using the distributed firewalls technique by letting the manager play the role of a property certification authority who issues property certificates to the group members. The group policy enforcement is then achieved by the group manager challenging and verifying the property attestation with each member of the VO.

For example, upon satisfaction of an attestation according the VO security policy and the remote site policy, the manager could release a group session key to the attested remote environment and this group session key plays the role of the "security association" (in IPSec language) for that entity to penetrate the distributed firewall (*i.e.*, to secure each packet both in data integrity and in message confidentiality). Thus, conference discussions in this environment can be securely conducted.

7 Trusted Computing Implementation and Deployment Status

The TCG has defined the security subsystems in such a manner so as to allow cryptographic applications to evolve easily from basic hardware protection mechanisms, such as key hardening, to more advanced capabilities, such as platform attestation and key backup and recovery services. The TCG whitepaper "Writing TCG Enabled Trusted Applications" (at the "Downloads area" of [29]) provides an overview of the strategies that application developers may employ in developing TCG-aware client applications.

The TCG Software Stack (TSS) provides trust services that can be used by enhanced operating systems and applications. The TSS uses cryptographic methods to establish security services and trust relationships, allowing applications to maintain privacy, protect data, perform owner and user authentication, and verify operational capabilities of the platform.

The TCG Crypto Service Providers (CSPs) provide features that are commonly associated with cryptographic functionality. A TCG-enabled platform typically supports both PKCS#11 [26] and the MS Cryptographic API (MS-CAPI). If an application developer has experience writing with PKCS#11 or MS-CAPI, it is relatively easy to provide basic TCG enabled capabilities. For most applications, the application developer may harden RSA asymmetric private key operations by simply calling the new CSP that is provided with TPM-enabled platforms. While there may occasionally be a subtle user experience difference based on different vendors' TSS and CSP, the TCG organization is working to develop common interfaces and actions that may, over time, facilitate a common user experience, independent of the platform.

In order to utilize the enhanced capabilities of the TCG-enabled platforms, the application developer must use the SDKs provided by the TPM manufacturer or OEM to expose the advanced trustworthy capabilities. An application developer may take advantage of a trusted platform's attestation capabilities by modifying their applications to require and verify the proper credentials provided by an attestation server. Eventually, most of the TPM and platform vendors will support the necessary credentials for attestation to function properly. Interoperability and compliance testing is being put in place and all the platform vendors have committed to supporting this mandatory aspect of the TCG specifications. Attestation servers are available from multiple vendors, including Verisign and Wave Systems, and some of these server products can assist in bridging the capability requirements of the platform's current limitations.

TCG-enabled PC platforms with TPM version 1.1b, both in desktop and notebook machines are now widely available from several computing systems manufactures. These include Dell, Fujitsu, HP, IBM and Intel (TCG "Fact Sheet," available at the "Downloads area" of [29]). These commercial-off-the-shelf products offer key storage for securing users' cryptographic credentials.

7.1 Known Challenges

As noted by [8] and [9] the remote attestation envisaged above is disappointingly fragile. There are many elements contributing to the runtime environment of a given piece of code. Operating systems, dynamic libraries, virtual machines, configuration files, etc. may all be upgraded or patched, leading to an explosion in the number of environments to be certified. In a realistic production grid, this will certainly be the case. Although we may hope to limit the scope of this heterogeneity as much as possible (because other behaviors may change as a result of differences, not merely security properties) the number of likely variants is probably too great to manage. A benefit of the Grid environment is the notion of a Grid Information Service (GIS), which might reasonably hold information about system configuration, and — if trusted — could hold relevant attestation information also.

Haldar *et al.* [8] propose *semantic attestation* wherein a "Virtual Trusted Machine" is attested using the TPM mechanisms, and then the programs running upon the virtual machine — Java or .NET perhaps — are attested by their *behavior* rather than their binary properties (so that semantically neutral changes may be made at any time).

Marchesini *et al.* [9] describe a case study in which three gross levels of change frequency are envisaged: the operating system kernel is "long-lived" and attested by the TPM mechanisms; intermediate software (in their case, the code of an Apache server) is dubbed "medium-lived" and perhaps certified by a CA for the sake of a community; and detailed software (web pages etc.) is "short-lived" and protected by an encrypted file system, with periodically-updated hashes covering its integrity.

Some combination of these features would seem ideal for a grid or web services context. We might determine that in a dedicated web services host, the environment up to the virtual machine is stable enough to offer TPM attestation; the individual services might be assured in other ways. Conversely, many grid applications will not run inside a virtual machine (although their controlling logic may) since they must exploit native processor performance as totally as possible — for these, other solutions will be necessary.

The challenge, then, for Grid and TC is to find means of integration which will support the significant components of Grid infrastructure in as seamless a manner as possible. It is necessary to support the whole lifecycle behavior: provisioning and commissioning grid nodes, deploying software, authorising users and (critically) groups to perform particular actions, and so on. Support for fine-grained mandatory access control will require integration with the authorization services discussed. Service descriptions will need to support the best that semantic grid services have to offer; grid information services will need to record configuration information for attestation purposes.

8 Concluding Remarks

As Grid security is becoming a more and more important topic, a number of problems remains untackled by the current Grid security solutions. We have identified group-oriented security and distributed system behavior conformance as among the essential requirements for Grid security while being indifferently supported by the current Grid security solutions. We have argued that trusted computing technology, thanks to its inherent properties of group-oriented security and system behavior conformation, can provide suitable solutions to the identified Grid security problems.

As we are still in an early stage of problem identification and solution search, the suggested approaches should be considered as initial input to substantial further investigations, which should include not only their plausibility, but also their alignment with the current Grid security solutions. Nevertheless, as hardware and software support for TC is gradually becoming available, it is timely to consider how such tools can be used to maximum effect in enhancing trust and security in Grid environments.

Acknowledgements

Greg Astfalk reviewed an early draft of this paper and provided insightful comments and suggestions. Graeme Proudler provided help with the TC technology background. Paul Vickers provided help with the Grid computing background. Nigel Edwards and Dirk Kuhlmann provided insights in the notion of secure virtualization.

References

1. Alfieri, R., Cecchini, R.L., Ciaschini, V., dell'Agnello, L., Frohner, A., Gianoli, A., Lõrentey, K., Spataro, F.: VOMS, an authorization system for virtual organizations. In: Fernández Rivera, F., Bubak, M., Gómez Tato, A., Doallo, R. (eds.) Across Grids 2003. LNCS, vol. 2970, pp. 33–40. Springer, Heidelberg (2004)
2. Anderson, D.P.: BOINC: A system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh, PA (November 2004)
3. Anderson, R.: TCPA/Palladium frequently asked questions (2003)
4. Arbaugh, B.: Improving the TCPA specification. IEEE Computer, 77–79 (2002)
5. Chadwick, D.W.: RBAC policies in XML for X.509 based privilege management. In: Proceedings of SEC 2002 (2002)
6. Garfunkel, T., Rosenblum, M., Boneh, D.: Flexible OS support and applications for Trusted Computing. In: The 9th Hot Topics in Operating Systems, HOTOS-IX (2003)
7. Goldberg, R.: Survey of virtual machine research. IEEE Computer Magazine 7, 34–45 (1974)
8. Haldar, V., Chandra, D., Franz, M.: Semantic remote attestation — a virtual machine directed approach to trusted computing. In: VM 2004. USENIX (2004)
9. Marchesini, J., Smith, S., Wild, O., MacDonald, R.: Experimenting with TCPA/TCG hardware, or: How I learned to stop worrying and love the bear. Technical Report TR2003-476, Department of Computer Science, Dartmouth College, Hanover, New Hampshire (December 2003)
10. Martin, A., Cook, C.: Grids and Private Networks are Antithetical. In: Chivers, H., Martin, A. (eds.) Workshop on Grid Security Practice and Experience, Oxford, UK (July 2004)
11. Safford, D.: Clarifying misinformation on TCPA (October 2002)
12. Saltzer, J.H., Reed, D.P., Clark, D.D.: End-to-End Arguments in System Design. ACM Transactions in Computer Systems 2(4), 277–288 (1984)
13. Stainforth, D., Martin, A., Simpson, A., Christensen, C., Kettleborough, J., Aina, T., Allen, M.: Security principles for public-resource modelling research. In: IASTED (2002)
14. Atkinson, B., et al.: Specification: Web Services Security (WS-Security), Version 1.0, April 5 (2002)
15. Bair, R. (ed.), Agarwal, D., et. al (contributors): National Collaboratories Horizons, Report of the August 10-12, National Collaboratories Program Meeting, the U.S. Department of Energy Office of Science (2004)

16. Bellovin, S.: Distributed Firewalls. In: login: pp. 39-47 (November 1999)
17. Bolosky, W.J., Douceur, J.R., Ely, D., Theimer, M.: Feasibility of a service distributed file system deployed on an existing set of desktop PCs. In: Proceedings of International Conference on Measurement and Modelling of Computer Systems, pp. 34–43 (2000)
18. Foster, I., Kesselman, C.: Computational Grids. In: The Grid: Blueprint for a New Computing Infrastructure, Ch. 2, pp. 15–51. Morgan Kaufmann, San Francisco (1999)
19. Foster, I., Kesselman, C., Tsudik, G., Tuecke, S.: A security architecture for Computational Grids. In: 5th ACM Conference on Computer and Communications Security, pp. 83–92 (1998)
20. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications 15(3), 200–222 (2001)
21. Freier, A.O., Karlton, P., Kocher, P.C.: The SSL Protocol, Version 3.0. INTERNET-DRAFT, draft-freier-ssl-version3-02.txt (November 1996)
22. Globus Toolkit, http://www-unix.globus.org/toolkit/
23. ITU-T.Rec. X.509 (revised) the Directory — Authentication Framework. International Telecommunication Union, Geneva, Switzerland (equivalent to ISO/IEC 9594-8:1995) (1993)
24. Novotny, J., Teucke, S., Welch, V.: An Online Credential Repository for the Grid: MyProxy. In: Proceedings of the Tenth International Symposium on High Performance Distributed Computing (HPDC-10), August 2001. IEEE Press, Los Alamitos (2001)
25. Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S.: A Community Authorization Service for Group Collaboration. In: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks, p. 50 (2002)
26. RSA Security. PKCS#11 v2.20: Cryptographic Token Interface Standard. June 28 (2004), http://www.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf
27. http://www.serverwatch.com/
28. Thompson, M., Essiari, A., Mudumbai, S.: Certificate-based Authorization Policy in a PKI Environment. ACM Transactions on Information and System Security (TISSEC) 6(4), 566–588 (2003)
29. http://www.trustedcomputinggroup.org
30. Korpela, E., Werthimer, D., Anderson, D., Cobb, J., Lebofsky, M.: SETI@home - massively distributed computing for SETI. Computing in Science and Engineering 3(1), 78–83 (2001)
31. Stainforth, D.A., Aina, T., Christensen, C., Collins, M., Faul, N., Frame, D.J., Kettleborough, J.A., Knight, S., Martin, A., Murphy, J.M., Piani, C., Sexton, D., Smith, L.A., Spicer, R.A., Thorpe, A.J., Allen, M.R.: Uncertainty in the predictions of the climate response to rising levels of greenhouse gases. Nature 433, 403–406 (2005)

Innovations for Grid Security from Trusted Computing
(Transcript of Discussion)

Wenbo Mao

Hewlett-Packard Laboratories, China

Bruno Crispo: But why do you need to chain the certificates, I don't understand. Usually I look for, for example, storage, and then I go find somewhere that can provide the storage I need, but why do I need a chain?

Reply: Do you mean, you do it yourself?

Bruno Crispo: No, somebody else does it for me, but in a sense when I issue a request, it's to the grid, so everybody is authorised to respond.

Reply: OK, that view says, there is a server, which can do for any. In that standard architecture, you need some superbeing staying there to serve some other people, so we come back to the point, how big the superbeing is. So many people ask for him to serve, so you need to go back to the original idea that if you buy big servers, somebody has to buy them. Here in this model mostly you have your own, OK, you are bounded, and then you can carry on. In your approach, the user need not worry, but somebody needs to worry; here nobody needs to worry.

Marios Andreou: Sorry, I'm not quite clear, do you mean the policies of the local user are applied by default to the accessing user or not?

Reply: The accessing user is mapped to the local user policy, yes. After mapping you can propagate, you need to, so, these guys don't know their accessing user at all, since it's chained, they only know this guy.

Marios Andreou: So they're applying local policy to the accessing user?

Reply: Yes. There are different ways to say the same thing.

Bruno Crispo: Probably to audit the processes it's easier to use an all-software solution. I think there is a version of Solaris that allows the role of "internal auditor" to be set so as to audit also the activity of the administrator.

Reply: Is it hardware based?

Bruno Crispo: No, it is a feature of the operating system. The operating system also builds a log that is read-only even for the system administrator.

Reply: OK, that's nice to know.

Mike Bond: Are you building this on top of the 1.1b of the TPM specs, or on the 1.2 TPM?

Reply: For this, year one, phase one, 1.1, in future considering science collaboration in years four and five, we need to think about so-called attestation things.

Mike Bond: How many different people's identities are you considering putting in one TPM, you were saying that people can share a TPM, I mean, what sort of numbers are you talking about?

Reply: Virtually unlimited.

Mike Bond: But the TPMs that exist at the moment (OK, the API can be implemented in anything, but) the TPMs that exist and are deployed at the moment don't have a high level of tamper-resistance. If each of them only contains one person's credentials, then it costs a lot to hack a lot of people, but if you put all their credentials in one TPM, then you have a security risk.

Srijith Nair: Can't you virtualise the TPM?

Mike Bond: No, I'm talking about the physical device, because it costs a certain amount to open up a physical device, and then having so many virtualise that device.

Reply: For the client machine, this is not a big deal, for the server, TPM is only a name, you can use really IBM crypto processors.

Mike Bond: So do you have any plans to use the TPM API inside one of those, or your own API?

Reply: Indeed we do. This year's offering, it will be for the client mainly it's not a problem. For the servers still we are considering this is an experiment, and also companies are shaping the TPM servers, and we need to see how big these TPM servers are. But in any case, TPM uses this external storage with only some handlers protected inside TPM, and where most of the things are outside, of course it becomes slower, but especially for the client market, it's not a problem at all. Yes, the idea is that for in the future, I think, if you look at this room, everybody has a laptop so for the client things you don't share a lot.

Kenny Paterson: You should also bear in mind that the MyProxy approach already concentrates user credentials inside, so the trusted computing approach in some sense can be worse than what's already done.

Bruno Crispo: What you suggested is that the identity that actually has generated in the grid map should be from the TPM, right?

Reply: Yes, our TPM does the auditing process, and so you see here, the size really doesn't matter, it's a total, and then periodically the TPM can sign the result, and store it, it should be stored outside, in permanent, persistent storage, and then auditing will rehash everything.

The Man-in-the-Middle Defence

Ross Anderson and Mike Bond

Computer Laboratory, University of Cambridge
{Ross.Anderson,Mike.Bond}@cl.cam.ac.uk

Abstract. Eliminating middlemen from security protocols helps less than one would think. EMV electronic payments, for example, can be made fairer by adding an electronic attorney – a middleman which mediates access to a customer's card. We compare middlemen in crypto protocols and APIs with those in the real world, and show that a man-in-the-middle defence is helpful in many circumstances. We suggest that the middleman has been unfairly demonised.

1 Introduction

The man-in-the-middle is much maligned. The Security protocol literature abounds with middleman attacks, and designs for security architectures commonly assume that if we could just cut out any possibility of interception, so that endpoints talk directly and securely, then everything will be OK. This could not be further from the truth. More often than not, the party that cheats you *is* the very one you thought you wanted to talk to in the first place, rather than some large-eared villain in the shadows.

In real life the middleman is often an ally who defends your interests; he is an essential part of going about normal business. We have our estate agents, our lawyers and accountants – even our priests – all acting as middlemen and representing our interests to those who might otherwise harm us. Resentment of the middleman usually only arises when he serves more than one master, or acquires too much independent power.

In this paper we argue that computer security should restore the middleman to his proper status. We describe several protocols where participants would benefit from being shielded from the actions of other participants. In fact there is already a body of literature in computer security covering composition problems, and if we can apply these ideas more broadly, we might learn how to engineer security protocols for multiple middlemen.

2 Electronic Commerce

Since 2005, British bank payment cards use the EMV protocols – a development known to the public as "Chip and PIN". Instead of reading a static number from a magnetic strip, a payment terminal supplies a customer PIN to a smartcard which verifies it and computes a MAC on the transaction, using a key it shares with the issuing bank. On casual inspection, this appears to be an end-to-end protocol; but the customer does not have a trustworthy means of entering his PIN into his card, or of checking the transaction details (payee and amount).

Consider a concrete scenario: You go for lunch at a small London restaurant and pay using your chipcard, unware that the restaurant is corrupt. You ask for the bill, and the waiter brings a handheld terminal to your table. Meanwhile, on the other side of town, his accomplice is loitering in a jeweller's store. The waiter sends an SMS message to his accomplice, who goes up to make a purchase. As you insert your own card into the waiter's terminal, the accomplice inserts a fake card into the jewellers. The waiter's sabotaged reader simply forwards the traffic from your card wirelessly to the card at the jewellers. You enter the PIN, thinking you're paying for lunch, but in fact you're buying the crooks a diamond!

We investigated the EMV specifications to determine whether such a 'middleman attack' was possible and practical. It is, and there seems to be no easy way to extend the EMV protocol to sort it out. It then occurred to us that if the merchant (or a corrupt merchant employee) could insert a relay device to monitor and forward the EMV protocol, maybe the customer could add her own middleman to do the same job, but with her interests in mind. An economic analysis of the problem is that the chipcard defends the bank's interests; the terminal defends the merchant's interests; but no party to the protocol defends the customer. What is the electronic equivalent of taking your lawyer along with you to the shop?

3 The Electronic Attorney

Our solution is to create an electronic attorney – a device that participates in the protocol whether the merchant and bank like it or not, which is paid for solely by the customer, and which acts only in her interests.

In the case of EMV, the protocol requires clear PIN entry by the customer and clear transaction entry by the merchant, supplies both to the chipcard, and if the PIN is correct the chipcard computes a MAC on the transaction data for transmission to the bank. The path from the terminal to the card can thus be mediated by a gadget that gives the customer a trustworthy display of the payee and amount, and can supply or withold the PIN to the card. It can also keep an independent audit trail.

A real-world implementation would be a small device about the size of a credit card, with chipcard contacts at one end and a chipcard reader at the other, as well as an LCD display and several buttons. The customer would place her chip card into it and then insert both devices into the merchant terminal. If the displayed payee and amount meet her expectations, she presses an approval button, releasing the correct PIN to the card, and writing an audit entry. Matters could be arranged so that the customer does not know the true PIN at all, and thus all transactions must be made via the attorney: this protects the customer against attacks by crooked merchants who skim the mag stripe and use that in an overseas ATM with the observed PIN. It also strengthens her hand in the case of a dispute: if the bank says ‘You must have done it, because our system says so and is secure’, she can retort ‘Not at all – my device is even more secure, as it’s evaluated to EAL4 unlike your 20 million lines of crufty old COBOL.’

4 Other Man-in-the Middle Defences

While there is some literature about shared-control processes (such as the privacy guard in CAFE [5]), the protocol community has not yet recognised the middleman as useful. There are of course middlemen in other security spheres: think of the firewalls and virus checkers that mediate between your PC operating system and the outside world. These middlemen’s job is to stay up-to-date with the latest threats and check incoming bits for signs of hostility.

Security APIs – the big brothers of security protocols – could well gain from middleman defences. The Hardware Security Modules (HSMs) at banks that perform PIN processing must conform to dated APIs with fundamental weaknesses in the encrypted data formats (described in detail in [1,3]). When an HSM must operate in a hostile environment (such as a semi-trusted facilities management firm, or a data centre in a unstable foreign country), an additional middleman is a logical solution to the problem. There apparently isn’t the economic incentive for first-world banks operating their own data centres to push for replacement of the API. But, where needed, a further device can act as a gateway to the banking HSM, observe and filter the transaction stream, stall transactions that look suspicious, and keep an independent audit trail of all activity.

Finally, we considered software middlemen for banking APIs in order to deal with short-term defence against the decimalisation table attack [2]. A hardware equivalent might involve mounting an HSM such as a 4758 inside a PC, and then mounting the PC in a steel box with a tamper-sensing barrier.

5 Composing Middlemen

In their influential paper “Cascade Ciphers: The Importance of Being First”, Ueli Maurer and Jim Massey showed that when ciphers are composed, the resulting cipher is as strong as the first, except in the case where the ciphers commute in which case the composition is as strong as the best. Defensive middlemen work similarly. If both my electronic attorney and an electronic attorney belonging to the Mafia are plugged between my chipcard and a merchant’s terminal, then so long as it’s my attorney that is next to my card, I will be all right; however, if a Mafia-owned attorney is in direct contact with my card, then it can perform arbitrary middleman attacks.

The case of commuting defences is more subtle. In the cipher case, this refers to stream ciphers; in the case of electronic attorneys, one can imagine a number of devices communicating with both card and terminal using a dependable broadcast protocol, so that any bad advice could be countered by denunciation. Something similar may be found in business, where a company thinking of a takeover might have a conference with multiple complementary specialists (bankers, lawyers, brokers, accountants) making suggestions and trying to find flaws in suggestions made by others.

6 Conclusions

The middleman has traditionally been seen as evil by security protocol designers, and attempts are made (often in vain) to exclude him. In real life, however, middlemen are ubiquitous, and we think the time has come for a rethink.

Designers like to aim at an elegant and incorruptible protocol for a broad range of tasks, but then fail for all manner of reasons, from unclear or dishonest assumptions, through shifting goalposts and featuritis to committee design. In the resulting complex, real-world protocols there is a place for a middleman. Consider the law: even a clever and articulate defendant retains a lawyer if he can afford it, since the complexity and volatility of legal protocols make it uneconomic for a nonspecialist to maintain the capability to argue a good case on his own behalf. Similarly, keeping up with computer viruses is a full-time job: no sensible security expert would maintain his own virus checker unless that were his speciality. In the case of EMV, it seems to be more by luck than by judgment that the protocols are open to middleman defenses. However, the economics suggest that they will stay that way. Tweaking the base protocols to make middleman attacks harder would be immensely expensive and take years to roll out, but keeping a middleman up-to-date is much cheaper.

To summarise, the man-in-the-middle defence is a good way to do two things. First, it is a sensible place to introduce a dynamic and upgradeable element which allows a slower but more careful evolution of an underlying protocol, or the retrofitting of protection to a protocol which is too expensive to change. Second, it gives us an opportunity to bring the human back into the protocol where there was no window for manual intervention before.

References

1. Anderson, R., Bond, M., Clulow, J., Skorobogatov, S.: Cryptographic processors – a survey, University of Cambridge Computer Laboratory Technical Report TR-641
2. Bond, M., Zielinski, P.: Decimalisation Table Attacks for PIN Cracking. University of Cambridge Computer Laboratory Technical Report TR-560
3. Clulow, J.: The Design and Analysis of Cryptographic APIs for Security Devices. MSc Thesis, University of Natal, SA (2003)
4. Maurer, U., Massey, J.: Cascade Ciphers: The Importance of Being First. Journal of Cryptology 6(1), 55–61 (1993)
5. Boly, J.-P., et al.: The ESPRIT Project CAFE – High Security Digital Payment Systems. In: Gollmann, D. (ed.) ESORICS 1994. LNCS, vol. 875, pp. 217–230. Springer, Heidelberg (1994)

The Man-in-the-Middle Defence
(Transcript of Discussion)

Ross Anderson

Computer Laboratory, University of Cambridge

The man-in-the-middle defence is all about rehabilitating Charlie. For 20 years we've worried about this guy in the middle, Charlie, who's forever intercalating himself into the communications between Alice and Bob, and people have been very judgemental about poor Charlie, saying that Charlie is a wicked person. Well, we're not entirely convinced.

Once you start to consider dishonest insiders, you have to change the whole threat model, and the rather simplistic assumptions and models that people used in the 1970s and early 1980s just don't run any more. Now it might occur to you that E-commerce is perhaps in the same state that protocol development was 20 years ago in the mid 1980s. E-commerce assumes that you've got three parties, the customer, the merchant, and the bank, but in real life the threat is usually not Charlie, the long eared individual lurking in the shadows. The threat is Bob, the shop that you go into and do business with – because the shopkeeper is bent. Sometimes it's the bank who's bent. How do you deal with this?

The card you may think is your card, but it isn't; the bank's terms and conditions say very clearly that it belongs to the bank, and even if belonged to you, you've got no say over the software in it. It would cost you quite a lot of money to dig that software out of the card and disassemble it, and find out what it actually does. So you're not represented via the card, the issuing bank is. Similarly the terminal represents the merchant's interests, or perhaps – if we're being slightly more careful – the interests of the merchant's acquiring bank. No-one represents the customer. But thanks to Mike Bond, and earlier work by Jolyon Clulow, Sergei Skorobogatov, and so on[1], pulling this protocol apart, we now know that we can fix that.

Meet my attorney. This attorney is a version zero prototype. The attorney that you will eventually be able to get, if someone takes this up and runs with it as a business, will of course be a lot smaller and slicker. It will be a dinky little device about this size, and it will have on it an LCD display and a little red button, and it will have the male part of a smartcard – which you can see here – and it will have the female part – which is the socket here. You put your chip and PIN card into the attorney socket, and you put the attorney into the terminal at Texaco, or whatever, and the LCD display says £34.99, or whatever is actually being asked from your card. Never mind the display on the merchant's terminal, that can say what it likes; but we don't care any more because now

[1] See position paper.

we have got a trusted path and a trusted display. So I put my card, with its little card condom – which is the phrase that Markus came up with for it – into the merchant terminal; and I see the amount; and I say, yes, £34.99, that's the amount of petrol I took at the pump; and I press the little red button; and then the device, which knows my PIN, enters the PIN into the card, thereby authorising the transaction, and also writes an audit trail entry. Whatever PIN I enter into the merchant terminal doesn't matter, four zeros, whatever, those bits are just thrown on the floor.

OK, so what happens then? Well, phantoms appear in my account, and I go to the bank and I say, "Oi, I didn't make those!" and they say, "You must have done because our system's secure" and I say, "No, hang on a minute, my system's even more secure than yours because I hired this smart young man, Peter Ryan, from Newcastle University, to formally verify it all, and it's EAL6+[2], right, and compare that with your twenty million lines of spaghetti COBOL? Forget it, pal!"

Tuomas Aura: What if someone steals your wallet, the wallet contains both the smartcard and your attorney, and nothing is required to use this because it is sufficient to press a button.

Reply: You will get a premium device with a thumbprint reader attached to it, so you will have to swipe your thumbprint, and it will check that you are alive before it will do the transaction. Once you can start using the middleman, not just as an attack but as a defence, you really have an opportunity to start fixing a lot of the stuff that's wrong with protocols. That's the fundamental point here. You want biometrics? You can have it, with the middleman. You're no longer locked down by the contracts that twenty or thirty thousand banks have made with each other, with VISA, with Mastercard, and with the suppliers. That is just too much momentum, you can't do any upgrades or maintenance on that kind of security.

Kenny Paterson: This is wonderful Ross, but why would a merchant or bank put your device in his system?

Reply: Well, the merchant doesn't mind; the merchant just cares about being paid. The banks might object because it undermines their deniability, but they might have problems on the public relations front if they tried to object publicly.

Bruce Christianson: But presumably this is not something under my personal control. The attorney must be provided by some third party?

Mike Bond: Where do you find a good lawyer, is the question!

Reply: Lawyers get their money from their clients, and so the lawyers have their incentives properly aligned. As EMV fraud transaction padding, redirection, and

[2] ISO/IEC 15408 Evaluation Assurance Level.

all the rest of it begin to climb, there will quite possibly be a market for people to build devices like this. Certainly, we know the electronics industry in China would be quite capable of producing these devices for a few pounds each: all you need is a small microcontroller and a relatively small and simple piece of software. The means for verifying small and simple pieces of software have been worked on for forty years.

Bruce Christianson: Oh sure, what I'm asking about is the threat model. The attorney isn't just your agent, he's not just a part of your end-system, he really is a man in the middle.

Reply: Absolutely, and if you have bent attorneys that can cause serious problems. Then there's the question of what happens when the merchant has his attorney, and you have your attorney, and the bank has his attorney, and you've got a whole room full of lawyers, and that's something I'm going to come on to shortly.

Bruce Christianson: There comes a point where the bank is going to say, well actually the fact that my customer had the benefit of an attorney strengthens my position. The question is, why is the bank going to allow, or encourage you, to plug an attorney into the card. Now I can see certain assumptions on the attorney, under which the banks are going to be only too eager for you to do that.

Reply: Well the banks cannot, as a physical matter, stop you plugging an attorney in, because of the way they designed the protocol. The only way they can stop you using an attorney is if they move from static data authentication to dynamic data authentication, but that would cost them money.

Mike Bond: There are still ways to use an attorney if you look inside the protocols. If they change the protocols enough then they can write it out, but even with dynamic data authentication you can still, for instance, send a PIN to the card. The card would have a private key on it, so you can send the correct PIN to the card, but you cannot piggyback, you can't decrypt things coming from the terminal. So for instance, if you wanted you could use your attorney to give you a one-time PIN, which you type at the terminal that doesn't encrypt it, the attorney could check it and pass on the real PIN. So there's a range of different attorneys, some of them would work with dynamic data authentication, some of them wouldn't.

Audience: But that doesn't stop the banks putting legal things in place to stop you.

Reply: That is why this is not just a technical paper, but also it's a challenge to the banking community: guys, are you honest or not? If you are dishonest, object to this paper and say that you will not let your customers use attorneys, if you are honest then say, that's fine. So please Mr NatWest, tell us, are you or are you not a crook? May I or may I not use an attorney when I am doing business with one of your point-of-sale terminals?

Frank Stajano: Maybe this is a silly question, but I can't understand why the terminal doesn't notice that you're not using the PIN from its own keypad.

Reply: Because that's the way the protocol is designed. The terminal sends the PIN to the card, and either gets back a MAC or it doesn't. So if my middleperson device, my attorney, takes the customer-entered PIN from the terminal and throws the bits to the wind, and then puts in its own PIN, the terminal has no way of knowing.

Matt Blaze: So you're basically adapting EMV to the Chaum Wallet model[3], where you have a device that represents the customer's interest acting as a wrapper around a device that represents the bank's interests?

Reply: We're aware of Chaum's work, but he's actually doing something rather different for his protocol.

Matt Blaze: Well fair enough, the implementation is different, I'm talking about the model, right, and essentially what you're doing is observing that EMV is susceptible to Chaum's model.

Reply: Chaum is representing only a privacy interest, and we're providing trusted path, trusted display, and audit.

Chris Mitchell: I think the banks would have a legitimate concern about you putting your device in the merchant terminal, because although you may not think it's a very secure scheme, to some extent the security relies on the merchant checking that the card is a genuine card. A malicious card – or a malicious sleeve – could generate a false MAC and give it to the terminal, and if the terminal doesn't do an on-line authorisation, it doesn't have any way of knowing whether the MAC provided for the transaction receipt is a genuine MAC or just a random string of bits.

Reply: So the local small businesses are all compelled to go and do an on-line check.

Chris Mitchell: But it does violate part of the security model the banks have.

Reply: But that's basically a niggle, that's the sort of thing that the bank would use as an excuse.

[3] Wallet Databases with Observers, Crypto '92.

Chris Mitchell: Well they would say that, but you would say that too.

Mike Bond: I can think of a compromise here which is, devices with card-retained functionality – that's where the card goes all the way in – you need quite an expensive, complicated attorney, with a short-range radio link. And even then if they choose to desire a card retained in such a way, they can make it very difficult for you to make a neat attorney with a screen and keyboard that you can still physically access whilst it's reading the card. So those things would make it difficult to use your attorney. Anything where you partially insert the card into a slot, such as shops and supermarkets, you probably could use the attorney.

Steven Murdoch: There are a lot of cases where the merchant never actually sees the card, never holds the card. If you look at the train station, the terminal PIN pad is too far away, it's behind a glass screen.

Mike Bond: The culture's changing in electronic payment now in the UK in that the merchant doesn't want to see your card any more. You do business with the box, and then the merchant just looks at the result from the box and says yes or no. You do business with something that's on the merchant's table but which belongs to the bank, and you don't really do business with the merchant any more. He just looks, says yes or no, and then gives you the goods.

Alex Shafarenko: But how are you going to prove to the bank that your attorney is not a spoof itself?

Reply: I don't have to prove anything to the bank. It's none of the bank's business.

Alex Shafarenko: Well, do you participate in the transaction if you're suspicious of me?

Reply: Well, for example, I refused to participate in an EMV transaction at Tesco's because when I put my card into the chip and PIN terminal, the girl snatched it away and said, "No, you mustn't do that, I must do this." Then she swiped my card through a mag stripe reader into a chip and PIN dock. Right, that means that Mr Tesco has got my mag stripe data and my PIN, and it went in clear text through computers that he programs. I said, "No, I am not putting my PIN into this terminal," and I paid cash instead.

So the attorney is not there to prove anything to the bank. It's there to prove to the judge – a big difference. I don't care about the bank.

Kenny Paterson: Since we're all having such fun do you mind if I sketch a phishing attack on your system. So the phisher sends out a whole lot of bogus sleeves containing a dodgy program?

Reply: Oh absolutely, then you're dead.

Kenny Paterson: Yes, so that does lead to the observation that this only works if you can trust the supplier of your device.

Reply: Absolutely, so you must initiate the purchase. You must go down to RadioShack, or whatever, and say, I want a sleeve that carries a guarantee, and the people who sell this must actually sell it with a guarantee, you know, "We have got EAL3+," or whatever they reckon is enough certification for it.

Kenny Paterson: Then my recourse is to the insurer.

Reply: And presumably you sell this as part of a package of fraud assurance mechanisms, whereby you say, "If you use our device and evil happens to you, then we will provide you with ten thousand pounds' worth of legal assistance to sue the bank."

Kenny Paterson: I'm sure we'd have great fun cooking up some scare stories.

Bruce Christianson: This is not unlike the old version of a guardian. If you've got a piece of hardware that belongs to someone else, and you're required to attach it to a network in order to do a transaction, you don't attach it directly, you put a firewall on a bridge, which you control.

Reply: Well exactly, this is what we come to with this slide.

Bruce Christianson: As an extension of that principle, it's very sound.

Reply: Right, mail guards, anti-virus software, firewalls, and so on, we've all got...

Matt Blaze: When I said this, you yelled at me. [laughter]

Reply: I thought you were in effect asserting that I'm infringing David Chaum's patent, and I'm saying, it's insufficiently similar to constitute an infringement.

Matt Blaze: Ah, I actually said it's similar to the model.

Reply: So there's the dining attorneys sitting round the table objecting to each other's schemes, and this is what you can use in a firewalling environment.

Also, it's the place to put the human back into the protocol, which is the theme of this workshop. Too often stuff is designed so that the person who owns the kit is designed out of the loop. Designing the bank customer out of the loop is absolutely classical of the way that techies do stuff: the bank is paying you your consultancy money, so you think about the bank's interest. Every banker knows that people are more likely to change their spouses than to change their bankers, so who cares about the customer? But all of this adds up; eventually it tips. We've got to have ways of putting the human back into the protocol, as well as getting the bugs out.

So what's rehabilitating Charlie about? It's about usability, and it's about maintainability, which I reckon are likely to be the two big problems in security engineering over the next five years. Voilà.

Tuomas Aura: I just want to point out this attorney that wants to give the PIN directly to your credit card, does not communicate with any other terminals, or any other middlemen?

Reply: He shouldn't, because if you have him in a WAN telling your PIN to the Mafia attorney, and the KGB attorney, and the North Korean attorney, and so on, that's bad stuff.

Tuomas Aura: So it may well be that you cannot compose this attorney, because all of them have special requirements that are not compatible.

Reply: Exactly so, I know. But my point is that the Maurer-Massey[4] insight gives us a conceptual framework in which you can explore when things should commute, and when they definitely shouldn't.

Mike Bond: I think what Tuomas is getting at is, what modifications of the EMV protocol are feasible, and which ones aren't. Is this really a protocol which can be hammered into a new shape, rather than being redesigned, and even deployed. And the answer is, with static data authentication, you would have a very hard time making multiple middlemen work usefully. Maybe with dynamic data authentication, where every player has a private key, it's a bit more feasible, but you still have the denial of service problem.

Reply: And then you've got the terrible problem of knowing which is the genuine terminal. You may be able to put your PIN in, encrypted under a terminal's public key, but one of the middlemen says, hello, I'm a terminal, and another says, hello, I'm a terminal. The Mafia own a bank so they can certify a terminal, so their certificate looks as good to the EMV system as the genuine terminals.

Bruce Christianson: In a weird kind of way things get better once the EMV protocols are redesigned with middlemen in mind.

Reply: Embrace and extend.

[4] Cascade Ciphers: The Importance of Being First, ISIT '90.

Using Human Interactive Proofs to Secure Human-Machine Interactions via Untrusted Intermediaries

Chris J. Mitchell

Information Security Group, Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
c.mitchell@rhul.ac.uk
http://www.isg.rhul.ac.uk/~cjm

Abstract. This paper explores ways in which Human Interactive Proofs (HIPs), *i.e.* problems which are easy for humans to solve but are intractable for computers, can be used to improve the security of human-machine interactions. The particular focus of this paper is the case where these interactions take place via an untrusted intermediary device, and where the use of HIPs can be used to establish a secure channel between the human and target machine. A number of application scenarios of this general type are considered, and in each case the possible use of HIPs to improve interaction security is explored.

1 Introduction

There are many situations in which a user is forced to conduct transactions employing a user interface which he/she does not entirely trust. Examples include the following.

– Many smart card applications require the user to insert his/her card into a terminal which does not belong to the user. In such a case, the user must trust the terminal both to correctly convey his/her instructions to the card, and not to make a record of any confidential information that is transferred. For example, if the card is being used to authorise a transaction, then the user must trust the terminal to instruct the card to authorise a transaction of the correct value.
– When a user employs a 'public' PC (*e.g.* in an Internet café or an airport terminal) the user has no guarantees that the terminal is not manipulating his/her instructions. For example, if such a terminal is used for a purchase, the user has no way of verifying that the terminal is correctly ordering the goods required.

This is a well-known and important practical problem, but one which is also difficult to address. Previous work has considered various aspects of this problem — see, for example, [1,2,3].

In this paper we consider a new approach to this problems, based on so-called Human Interactive Proofs (HIPs). In particular, we consider a number of different scenarios in which humans and machines (*e.g.* smart cards, remote PCs, etc.) communicate via untrusted intermediaries. In each case we also consider how HIPs might be be used to improve security for such an interaction.

2 The Main Idea

2.1 Human Interactive Proofs

HIPs, as discussed, for example, by Dhamija and Tygar [4] 'allow a computer to distinguish a specific class of humans over a network'. We are interested here in the simplest case, *i.e.* where humans are distinguished from computers. Again, as specified in [4], 'to do this, the computer presents a challenge that must be easy for ... humans to pass, yet hard for non-members [*i.e.* computers] to pass'.

Probably the most common example of a HIP in use today is where numbers or letters are displayed in a distorted fashion so that only humans can read them. However, many other techniques have been proposed, *e.g.* to choose a pair of images showing the same person (from amongst a larger set), or to count objects of a particular type in a scene [5].

Many applications for HIPs have been proposed. Probably the best known is the use by Yahoo (`www.yahoo.com`) to restrict access to free email accounts. Here a user must solve a HIP before obtaining such an account; this prevents automatic harvesting of email addresses for use by spammers.

2.2 Establishment of a Secure Channel

We now consider the possibility that HIPs could be used to establish a secure channel between a human user and a trusted device when all communications take place via an untrusted intermediary. Of course, this will not be a perfect solution to all the security problems, but it might significantly reduce threats in some situations.

It is not hard to see that, in some sense, a HIP can provide a low bandwidth communications channel between the computer and the human, which no other *computer* can intercept. In the case of a HIP based on distorted characters, the computer could display a message which the human can read but no other computer can understand. As a result, the HIP provides a confidentiality-protected but unauthenticated communications channel from the computer to the human.

We can also use HIPs to provide a confidential but unauthenticated reverse channel, *i.e.* a communications channel from the human to the computer. One such approach would involve the computer displaying a sequence of distorted messages (unreadable to another computer), of which the user chooses one (or more). This could, for example, and as discussed below, enable a user to enter a secret password or PIN into a computer in a fashion unobservable to a computer, even if can monitor all the data transfers. More generally, for secure password entry, the computer could ask a password-related question (in some distorted form) and receive an answer via the user selection of one of a series of distorted images.

The use of HIPs therefore allows the establishment of a two-way confidential channel between the user and a computer, even when communications pass via an untrusted intermediary. Some examples of possible uses of such a HIP-protected communications channel are sketched in the following sections, although these ideas need further analysis. There may also be further scenarios in which these ideas have value.

Before proceeding we also note two fundamental limitations of the communications channel established using a HIP.

– Firstly, the channel is confidentiality-protected but not authenticated. This means that there is the risk of man-in-the-middle attacks. Overcoming such attacks probably means that techniques of the type we have just described may need to be combined with other techniques for human authentication of remote machines. Of course, in principle it is not difficult to transform a confidential channel into an authenticated channel, given the two parties possess a shared secret. However, the problem here is that the user does not necessarily have any means to perform cryptographic computations.
– Secondly, if a human interceptor is present, then the channel is no longer confidentiality-protected. That is, the technique only offers protection against automated attacks.

3 Scenario 1: Smart Card PIN Entry

When a smart card is inserted into a terminal for some purpose, it is often necessary to insert a PIN to authorise a transaction (this proves to the card that the cardholder is present). If the terminal is untrusted by the cardholder, as is often the case in practice, then there is a risk that the PIN will be immediately and automatically compromised. To try to reduce this risk, the card could use a HIP to prompt for the PIN, *e.g.* by displaying a distorted numeral by each button on the terminal, and inviting the user to type the button corresponding to the first PIN digit, then the second digit, and so on.

Of course, this would require the card to provide the distorted pictures to the terminal, which would then display them. The terminal would then send back to the card the identifiers for the images selected by the user. If the terminal displays the images as requested by the card, the terminal will not learn the PIN, *i.e.* this use of a HIP allows PIN entry via an untrusted terminal to take place in such a way that the PIN cannot be automatically captured. However, if a human monitors the PIN entry process, either in real time *or subsequently*, then the PIN is revealed. This nevertheless makes it more difficult for malicious entities to collect large numbers of user PINs, since the process cannot be completely automated.

Interestingly, the above described process does enable a malicious terminal to learn the PIN if it fails to follow the protocol correctly. That is, if it displays its own images (for which it knows the corresponding digits), instead of the ones provided by the smart card, then it can immediately and automatically learn the correct PIN. However, such an attack would not enable the terminal to enter the PIN it has learnt into the card (at least, unless a human was involved). Hence, the transaction would not be completed. This issue is, of course, a direct corollary of the fact that the HIP-based communications channel is not authenticated.

These latter observations raise the possibility that security might be further improved by incorporating the use of the visual authentication ideas of Dhamija and Tygar [6]. That is, the images produced by the card might be personalised to the user, and hence the user will detect when the terminal displays its own images to try to capture the user PIN.

However, even if this latter approach is followed, the PIN will still be revealed if a human monitors the procedure, either at the time of the transaction or later. That is, the main benefit of the process is in preventing automatic collection of card PINs. It would be interesting to see if the process could be further enhanced to offer a greater degree of security. Possible extensions to and generalisations of the above process include using the same process to protect the transfer of PINs (or passwords) to a remote application via an untrusted terminal. This is discussed further in Scenario 3.

4 Scenario 2: Smart Card Transaction Authorisation

As mentioned in the previous section, smart cards are often used in terminals that are not trusted by the cardholder. In particular, the cardholder may use the card to authorise a transaction, and the card will authorise the value of the transaction on behalf of the cardholder (*e.g.* by signing an authorisation message containing this value). The risk here is that a malicious terminal could change the value of the transaction, since the cardholder has no way of knowing exactly what message the card is authorising.

In a similar way to that described in the previous scenario, the card could ask the cardholder to enter the transaction value by prompting using distorted pictures of digits. That is, the terminal would not know how to enter the correct (or any specific incorrect) transaction value into the card. The card could then display a distorted image of the total transaction value for the cardholder to verify. Finally, once this interaction has completed, the card will provide the authorised message to the terminal. As in the case described in the previous scenario, the security of the process might be improved by using HIP images tailored to a particular user, so that the cardholder will be able to distinguish between images presented by a fraudulent terminal and those presented by the card. That is, prior shared secrets can be used to provide a measure of authentication for the channel.

Of course, as in all cases considered here, if a malicious human can monitor and interfere with card-cardholder communications at the time of the transaction, then the use of a HIP does not help. However, unlike the previous scenario, if a human only has access to a transaction record after the event, then there are no obvious attack strategies. That is, the use of a HIP in this scenario appears to offer greater benefits than in the previous case.

5 Scenario 3: Using an Untrusted PC for a Remote Login

The third scenario we consider here involves the use of an untrusted terminal, *e.g.* a PC in an Internet café, for remote login, *e.g.* to a server operated by the user's employer. This is a common practical scenario.

Suppose, moreover, that a user employs a hardware password-generating token when performing remote logins. Such tokens, that generate 'one-time passwords' valid for only a short time period, are in wide use, and are designed to reduce the risks of sending fixed passwords via intrusted intermediaries. However, if the terminal is fraudulent, then it could potentially copy the one-time password, and use this to login other terminals to the remote computer (as long as the logins occur within a short space of time).

The use of HIPs to secure the communications between user and remote computer might help alleviate this problem. The same strategy could be used as that described in Scenario 1, *i.e.* where the user is invited to enter the one-time password via images displayed on the terminal.

Moreover, given that the password has only a short lifetime, the possibility that it could be learnt by a human monitoring a recorded transaction later, is no longer a significant threat. That is, the security situation is improved because of the short-term nature of the secrets transferred across the channel.

6 Scenario 4: Using an Untrusted PC for E-commerce

In an electronic commerce transaction made using an untrusted terminal, the user is typically required to enter account details to complete the transaction. These details would typically constitute not only the user PAN (Personal Account Number), but also the three-digit security code designed to reduce the risk of re-use of stolen account details. (To reduce the risk that these three-digit codes are themselves stolen, merchant servers are prohibited from storing them).

This type of transaction therefore poses a significant security risk to the secrecy of the cardholder account information. As in Scenarios 1 and 3, HIPs might be used to protect this end user information against automated capture. The analysis would appear to be very similar to that in Scenario 1.

7 Scenario 5: Context Establishment

Our final scenario is a little different. It may be possible to use HIPs to increase the security level of a communications channel used for initial security context establishment. One scenario in which this idea might be useful is 'near-zero-configuration' of personal devices, *e.g.* in the home. This relates, of course, to Stajano and Anderson's notion of imprinting of devices (see, for example, [7]).

Currently, protocols such as Bluetooth involve a pairing process, which, as currently specified, is rather insecure. (More secure alternatives exist, but have yet to be implemented in Bluetooth — see, for example, ISO/IEC 9798-6 [8], or section 10.7 of [9]). It would be interesting if HIPs could be used to enhance the security of such initial exchanges. Of course, the use of a HIP would not help against an active human interceptor, but might help against the more likely threat of an eavesdropper capable of automated interference with a protocol, but not capable of breaking the HIP.

The scenario of use envisaged is where a naive user has to initialise a device, but does not have the expertise (or the patience) to engage in a sophisticated security initialisation procedure. Moreover, if the devices being 'paired' lack a user interface, then the more secure processes described in ISO/IEC 9798-6 are difficult if not impossible to use. We therefore consider the case where the devices being paired make use of untrusted third party devices which do possess the necessary sophisticated user interface.

More specifically suppose that a PIN-based procedure is to be used to secure device pairing, where the PINs are entered into the devices being paired via third party (untrusted) devices. In such a case, if no additional security measures are taken, the third party devices could subvert the pairing process, almost regardless of how the pairing protocol actually works. The use of a HIP, *e.g.* where PIN entry involves selection of distorted images of characters, might enable this pairing process to be made secure, at least against machine-only adversaries.

If the pairing process involves a Diffie-Hellman key exchange authenticated using PINs, then it may be possible to design the scheme so that a human adversary involved only after the exchange cannot break the established key. That is, the security context established between the two devices will remain sound. However, the details of such a system remain to be worked out in detail, and will, of course depend on the details of the operational scenario.

8 Conclusions

We have conducted an initial investigation of a variety of scenarios to see how HIPs might be used to improve the security of human-machine interactions via untrusted intermediaries. The preliminary conclusion would appear to be that, whilst HIPs can offer some benefits in all scenarios, the greatest benefits occur in scenarios where the interaction involves the completion of a transaction (*e.g.* a purchase of goods, or the establishment of a security context), rather than merely the transfer of a password or PIN, unless the password or PIN is only of short term interest.

This is because the use of a HIP does not prevent a human learning any secrets transferred in a monitored exchange either at the time of the exchange or later. However, if a process is completed during the interaction, the only way in which this can be attacked is if the malicious human monitor is actively interfering at the time of the exchange.

Note that it is sometimes much easier for a fraudulent individual to monitor interactions after they have occurred, rather than at the time of the interaction. Apart from the obvious convenience issues for the attacker, in some cases it may only be possible to examine interactions after the event, *e.g.* when key-loggers are used.

We would also point out that the scenarios introduced here have not been analysed in very great detail, and that there may be other more effective approaches for the use of HIPs. Given the significance of the underlying problem, this therefore appears to be an area meriting further research.

Finally, the nature of the secure channel which can be provided using a HIP would also appear to merit further investigation. In particular, one might reasonably ask which types of HIP offer the possibility of the highest bandwidth channels between human and computer.

Acknowledgements

The author would like to gratefully acknowledge a number of helpful comments made by participants in the 2006 Security Protocols Workshop — these have significantly improved the paper.

References

1. Berta, I.Z., Buttyan, L., Vajda, I.: Mitigating the untrusted terminal problem using conditional signatures. In: Proceedings of ITCC 2004, The International Conference on Information Technology: Coding and Computing, Las Vegas, NV, USA, April 2004, pp. 12–16. IEEE, Los Alamitos (2004)
2. Gobioff, H., Smith, S., Tygar, J.D., Yee, B.: Smart cards in hostile environments. In: Proceedings of the Second USENIX Workshop on Electronic Commerce, Oakland, California, November 1996. USENIX Association, Berkeley (1996)
3. Stabell-Kulo, T., Arild, R., Myrvang, P.H.: Providing authentication to messages signed with a smart card in hostile environments. In: Proceedings of the USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA, May 10–11. USENIX Association, Berkeley (1999)
4. Dhamija, R., Tygar, J.D.: Phish and HIPs: Human Interactive Proofs to detect phishing attacks. In: Baird, H.S., Lopresti, D.P. (eds.) HIP 2005. LNCS, vol. 3517, pp. 127–141. Springer, Heidelberg (2005)
5. Lopresti, D.: Leveraging the CAPTCHA problem. In: Baird, H.S., Lopresti, D.P. (eds.) HIP 2005. LNCS, vol. 3517, pp. 97–110. Springer, Heidelberg (2005)
6. Dhamija, R., Tygar, J.D.: The battle against phishing: Dynamic security skins. In: Symposium On Usable Privacy and Security (SOUPS) 2005, Pittsburgh, PA, USA, July 6-8, pp. 77–88. ACM Press, New York (2005)
7. Stajano, F.: Security for Ubiquitous Computing. John Wiley and Sons Ltd., Chichester (2002)
8. International Organization for Standardization Genève, Switzerland: ISO/IEC 9798–6: 2005, Information Technology — Security techniques — Entity authentication — Part 6: Mechanisms using manual data transfer (2005)
9. Dent, A.W., Mitchell, C.J.: User's Guide to Cryptography and Standards. Artech House, Boston (2005)

Using Human Interactive Proofs to Secure Human-Machine Interactions via Untrusted Intermediaries
(Transcript of Discussion)

Chris J. Mitchell

Royal Holloway, University of London

It's ironic that these hard problems, such as character recognition, have been known to be hard for a long, long time, and yet almost as soon as people make crypto things out of them, they get solved. Actually it's not quite the way you think because what's happened is that for the examples that get automatically generated, there are special techniques which work just because they've been created deliberately. It's not trivial to produce things that are really hard to solve, and some of the ideas for distorting characters have been quickly broken, but I believe there are some around which are quite robust.

Frank Stajano: Are they written by undergraduate students?

Reply: Yes, that's right, I guess you want things that are automatically generateable, because that's really nice. And there are other techniques apart from distorted characters, I've seen examples where you get six pictures and you're asked to choose which ones of these six show the same person, and they might show several people, but you have to spot that Tony Blair is in three out of those six, and these are the kinds of things that are quite hard for computers to do, or to count objects, you know, count cars in this picture.

Marios Andreou: What about speech, is that feasible, like audio to discern a voice pattern against noise, or something?

Reply: Maybe. You want things that are quite easy for computers to set, these might be quite hard for computers to set, although you could give a computer a large library of photographs with some rules about how to use them.

Jeff Yan: Actually, some CAPTCHAs implement this.

Reply: This talk really stems from the observation that we can use a HIP as a kind of communications channel between the computer and a human, which no other computer can intercept. So if we're using these characters the computer can display a message to the human that the human can read but no other computer can read. So we've got a confidentiality protected channel, and of course it's not very confidentiality protected because human beings who are monitoring the channel can also read it, but not machines, which is kind of nice because, to take bank card examples, these malicious attorneys, or whatever one might call them, that might get left in card readers, are typically machines, they're there to make a record of what happened. They can make a record perhaps of the images, but they can't actually read the data in real time, they need a human help to do that. And we can generate also a reverse channel, which is also confidentiality protected, but not authenticated.

So, in principle at least, with HIPs we can get a two-way confidential channel between the user and the computer, even when the communications pass via an untrusted intermediary but the channel is unauthenticated, it's also only confidential against machine eavesdroppers. But I claim that's a potentially useful property.

I've already hinted at the first scenario, where the PIN is sent to a smartcard via a confidential channel, by displaying concealed digits next to the buttons. The problem is, a bad man-in-the-middle could put up its own images, the bad terminal in the middle generates its own distorted images and learns the PIN, because it knows what the images mean that the user is now selecting. But it can't tell the card the PIN because it can't talk to the card, because the card is displaying its own distorted images, which the terminal can't read. But this doesn't offer much security because the first time the card asks for the PIN, the terminal displays its own images, learns the PIN, and then tells the user, oh that PIN failed, and then lets the card prompt for the PIN, so the terminal can learn the PIN, and allow the card to get the PIN, but at the cost of forcing the user to enter the PIN twice.

Ross Anderson: A sensible implementation would see to it that the images that the smartcard puts out would be customised to the user. It's also worth remarking that, again, one of Ueli Maurer's theoretical contributions[1] in 1996, was that you can get an authenticated channel from a confidential channel in general, but it also goes the other way, because if the smartcard sends me some random nonce on this machine confidential channel, and I type that random nonce, I have authenticated by existence.

Bruce Christianson: Or you can go a little bit better than that, the nonce could be a weak key that you then use to do an encrypted key exchange[2]. Now you have a strong key, and the human learning the secret after that point, is no threat.

Ross Anderson: Well my mental arithmetic isn't quite enough to do EKE.

Bruce Christianson: Oh well presumably you've got some PDA, or something like that, that can do it for you?

Ross Anderson: In that case I'll just use it as a card number.

[1] Journal of Computer Security **4**, 1: A Calculus for Security Bootstrapping in Distributed Systems.

[2] "Secure Sessions from Weak Secrets" LNCS 3364, page 190.

Mike Bond: I think there is limited crypto that can still be done in your head; some of the stuff that George Denezis and I talked about last year for secret society protocols would like you to be able to XOR very short numbers[3]. Things like that can be done, though maybe not by everyone.

Reply: Of course you also have the problem that if the machine records the transaction, it can play it back to a human later, and the human will assist the machine to learn the PIN.

Ross Anderson: We did have a system in COPAC which pre-dated EMV by several years, whereby your receipt also contained a MAC of the transaction, which was computed by your card, using a different key from the MAC that was sent off to the bank, but which key was also known to the card issuing bank, and could be produced for third party verification in the event of a dispute. It was interesting to see that this function, as it benefited the customer solely, and the bank not at all, was dropped from the EMV specification.

Mike Bond: I've got a suggestion for usage.

Reply: Oh, good.

Mike Bond: Well not really, it's actually an evil usage scenario. HIPs, and that sort of technology, is I think already deployed wide-scale by spammers for sending things that can't be read by machine, an image of, buy Viagra, and a machine doesn't know what it is, so it has to send it to the human to be read to see if it's spam or not.

Reply: So that's, in a sense, an application of what I'm doing here, but an evil application.

James Heather: It seems to me that there is a danger here because usually when you do something like RSA crypto, you're banking on a fairly well attested Moore's law, to protect you against advances in the technology, and algorithm design, unless somebody is going to break RSA, which looks fairly unlikely. But here you're relying on nobody coming up with something that manages to handle the sorted images, which, as far as I'm aware, could happen any moment, it's just that no-one's really worked out how to do it yet.

Reply: That's true. Of course you could say that about AES.

James Heather: You could, yes, but it seems intuitively less likely.

Reply: Yes, I agree. The first few years of the history of CAPTCHAs has been a sorry tale of broken schemes, but there are some which are still believed to be robust.

[3] "The Dining Freemasons" LNCS 4631, page 258.

Bruce Christianson: I do like Virgil Gligor's idea that you should make the CAPTCHA related to some hard problem in AI, because that way you get a good publication out of it either way.

Reply: I think the problem is, where they are known hard problems, we don't construct general instances of the hard problem, we construct special cases, a bit like knapsack crypto. Knapsacks are very hard, but unfortunately the instances we've created were easy ones.

Ross Anderson: I took your hint and looked up CAPTCHAs in Wikipedia, and about two thirds of the Wikipedia entry consisted of criticisms of CAPTCHAs and accessibility. For example visually impaired people are of course affected, and there's still thirty five thousand deaf/blind adults in the USA, so it looks like there are some social policy issues.

Bruce Christianson: On the other hand if you're colour blind, it's relatively easy to invent a CAPTCHA that only you will be able to do, or at least only people who suffer from your kind of colour blindness.

James Heather: Or use shade, which can simulate colour blindness very easily.

Bruce Christianson: Only if it can solve the underlying problem.

James Heather: Yes, OK, but your underlying problem is then, give me your security.

Ross Anderson: Well that depends on the threat model.

Bruce Christianson: Yes, it depends on the level of authentication that you require.

Ross Anderson: Presumably a mature technology admits customisation so that if you're red/green blind then the CAPTCHA you get at the cash machine will not be solvable except by someone with the same condition.

James Heather: You can put your disability here.

Ross Anderson: And related language for the audio components.

Frank Stajano: Well saying the CAPTCHA in Sanskrit is probably almost sufficient obfuscation.

Matt Blaze: So that suggests the biometric CAPTCHA, so for example, give you an eye chart and ask you to enter the first line you can read.

Ross Anderson: And you get somebody's age by going up the audio scale until he can't hear it anymore.

Reply: Yes.

Ross Anderson: Well hang on, you could use audio masking effects. I wonder if you could do this, I wonder if you could produce an audio signal that would be interpreted differently by a 21 year old and a 41 year old.

Srijith Nair: Just play contemporary music.

Ross Anderson: And you could play Mozart and make the 21 year old go away. It would be possible to use some signal processing trick so that you would actually hear different words spoken depending on your age.

Srijith Nair: Maybe if you used high frequencies.

Ross Anderson: Exactly, with a high frequency component which is tailored to change the performance. Well, a good research project for someone.

James Heather: You can certainly do a similar thing with images.

Srijith Nair: I thought something like that was done in Cambridge a couple of months ago. The intention was you could drive away kids, because kids can hear a particular frequency, but the grown-ups can't hear a particular frequency.

Ross Anderson: There's a guy in Britain who developed a teenager repellent[4] put near his shop, and then they announced it's unethical and he's not supposed to use it[5].

James Heather: You've got no protection with something like a human and a machine operating together, because they could strip out the high frequencies.

Ross Anderson: In that case you put something that's going to intercept the challenges.

Bruce Christianson: It depends a great deal on your threat model. Often what you want is a secure channel between the endpoints, but you don't know in advance who the individual at the endpoint is going to be. Often indeed that subsequent communications is going to happen over some other medium, and we just want the secure channel to establish the key for that. If you could have agreement about where the endpoint was, it's quite good because it means that for the attacker capturing the transaction, and getting a human involved later, it's too late. They can't go back into the agreement. But the hard problem is

[4] `http://en.wikipedia.org/wiki/The_Mosquito`

[5] Although just playing Mozart seems almost equally effective.

ensuring there isn't a false front-end, putting a camera on the screen and relaying it to someone else, and the poor user thinks that's the endpoint.

Ross Anderson: There's almost an isomorphism with these scenarios that David Wheeler was playing around with ten, fifteen years ago, where you used some kind of aid to enter passwords. Various people played with schemes like this, where you have a demonic aid that enables you to answer the challenge from the machine through an untrusted interface, and the bank-end stuff, it strikes me, is much the same in both cases. Such things are relatively trivial to design if they're going to be used one-off; they're much more problematic if the TSB gives the same card to each of its four million customers and they need to customise them all in manufacturing the ten by ten grid with different funny colours.

Bruce Christianson: If you really need to know which customer it is at that stage…sometimes you just want to know that you are speaking to a NatWest customer, and the customer wants to know they've got a secure channel before they enter their details.

Ross Anderson: Well I presume the obvious application of this is you can see that you're talking to a real access point, and nobody really knows how to solve that.

Mike Bond: In certain scenarios that you mentioned, say maybe a smartcard where you've got high bandwidth, you're not going over the Internet, could anything be done? Given it's a hassle for them to record an image, why can't it go up from images to full motion video? A smartcard may have a very limited power processor, but can it produce an output we could render at extremely low resolution on some kind of screen that renders something that can just be read by a human. It is a vast amount of data to store under the process, but a little scrolling message, or something, and that could be a race for tagging defence there.

Steven Murdoch: I recall something, probably a year ago, where they proposed that smartcards should render 3-D graphics, which would be even harder than sending CAPTCHAs.

Mike Bond: But I think when removing stuff it would be good. All a potentially evil device has to do at the moment is forward this display, you know, give me direct access to the inputs of anybody else in the display, and I'll make a colour and light show. So that should be what the terminal does now as a feature, it says, don't give me the amount and I'll display it, it's, right, I'm now talking serial to your LCD control.

Secure Distributed Human Computation
(Extended Abstract)

Craig Gentry[1,⋆], Zulfikar Ramzan[2,⋆], and Stuart Stubblebine[3]

[1] Stanford University
cgentry@cs.stanford.edu
[2] Symantec, Inc.
zulfikar_ramzan@symantec.com
[3] Stubblebine Research Labs
stuart@stubblebine.com

1 Introduction

In Peha's Financial Cryptography 2004 invited talk, he described the Cyphermint PayCash system (see www.cyphermint.com), which allows people without bank accounts or credit cards (a sizeable segment of the U.S. population) to automatically and instantly cash checks, pay bills, or make Internet transactions through publicly-accessible kiosks. Since PayCash offers automated financial transactions and since the system uses (unprotected) kiosks, security is critical. The kiosk must decide whether a person cashing a check is really the person to whom the check was made out, so it takes a digital picture of the person cashing the check and transmits this picture electronically to a central office, where a human worker compares the kiosk's picture to one that was taken when the person registered with Cyphermint. If both pictures are of the same person, then the human worker authorizes the transaction.

In this example, a human assists in solving problems which are easy for humans but still difficult for even the most powerful computers.

Motivated by this example, we suggest the notion of secure distributed *human* computation (DHC): a general paradigm of using large-scale distributed computation to solve difficult problems where humans can act as agents and provide candidate solutions. We are especially motivated by so called "AI-complete" problems which occur in fields such as image analysis, speech recognition, and natural language processing. Although DHC might sound far-fetched, several present-day situations exemplify this paradigm:

– **Spam Prevention:** Some spam prevention mechanisms [12,13,15] leverage human votes.
– **CAPTCHA Solutions:** Ironically, spammers can hypothetically use DHC to further their goal [1,2]. Consider free email providers who have incorporated special puzzles, known as CAPTCHAs, that are easily solved by humans, but challenging for computers, during the account creation phase to prevent spammers from automatically creating email accounts; spammers,

⋆ Part of this research was done while these authors were at DoCoMo USA Labs.

in turn, can farm these CAPTCHAs out to humans in exchange for access to illicit content.

- **The ESP Game:** In the ESP Game [3], two players are randomly paired over the Internet; they are not permitted to communicate, but both view the same image on their respective web browsers. Each player types in words that describe the image. As soon as both players enter the same word, they get a new image. Players get entertainment value and the game organisers now have labels for their images, which is valuable for improving image search.
- **Distributed Proofreaders:** Distributed proofreaders (www.pgdp.net) is a project that aims to eliminate optical character recognition (OCR) errors in Project Gutenberg (www.gutenberg.net) electronic books. A (small) portion of the image file and corresponding text (generated by OCR) is given side-by-side to a human proofreader who, in-turn, fixes remaining errors. By giving the same piece of text to several proofreaders, errors can be reliably eliminated.
- **Other examples:** Open source software development loosely falls into the DHC paradigm; here the difficult problem is not something crisp like image recognition, but instead that computers have a hard time automatically generating source code. As another example, consider Wikis, which are online encyclopaedias that are written by Internet users; the writing is distributed in that essentially almost anyone can contribute to the Wiki.

2 Applications to E-commerce and B24b

Web sites typically rely on three revenue sources: advertisements, subscription fees, and e-commerce. With respect to advertising revenue, one has to contend with issues like low click-through rates [5] and click fraud [14]. Also, outside of specific niche industries, few will pay subscription fees for premium Internet content.

However, DHC yields another revenue source: companies who want specific problems solved can farm them out to the hundreds of millions of Internet users. In exchange for solving the problem, some service or good is provided. We note that DHC payments have several advantages over credit cards. First, solving a human computation problem might be faster than fetching a credit card and entering the billing details. Second, credit card information can be compromised (*e.g.*, if the merchant web server is compromised). Finally, credit card transaction fees are substantial, so cannot be used for low-value content. In a sense, then, human computation can form a new type of online currency or bartering system.

As an example, such a mechanism might be useful on the New York Times web site (`www.nytimes.com`) which provides free access to the day's news articles, but charges a fee for archived articles. Such a fee (while necessary from a business perspective) might deter users – especially since they can probably (illegally) obtain the article text; *e.g.*, it was posted to a mailing list. However, instead of charging a fee, the New York Times could give the user a human computation problem (*e.g.*, transcribing an audio feed into text). In exchange for solving the problem, the archived article can be provided. This concept extends to other service offerings; *e.g.*, music downloads or long-distance minutes for solutions.

DHC may also enable the Business-to-Four-Billion (B24b) model [11] which aims to provide digital services (wireless communication, Internet, etc.) to the world's four-billion poorest people. Individually these people have annual incomes less than $1500 – yet they have large collective buying power. Although the economic feasibility of B24b is still very much an open question, providing services in exchange for solving DHC problems seems like a useful approach, since it depends on an abundance of human resources, while avoiding cash transactions. On the other hand, since we are talking about *human* computation, there are ethical issues to consider – in particular, as with any human service, we should ensure that the market for human computation is not unduly exploitative.

3 Related Fields

DHC is relevant to several research disciplines. With respect to information security, one can superficially view DHC as a type of secure multi-party computation (for a survey see chapter 7 of [8]), since it may involve multiple human computations, but perhaps the differences are more striking than the similarities. First, the parties are human beings instead of computers; second, the parties are themselves not providing actual inputs, but are instead providing candidate answers (which themselves can be construed as inputs into a group decision-making process); third, the "function" to be computed may not always have a clear-cut answer; fourth, the computation may be facilitated by a semi-trusted[1], but computationally "weak" server (*i.e.*, it cannot solve AI-complete problems itself); fifth, we may not always be restricted by privacy concerns, although they are important in a number of motivating applications.

To analyze security, we may consider the case where the adversaries are rational, and use game-theoretic tools. Also, since DHC is a form of currency, we may use cryptographic tools that have been developed in connection with e-cash. Finally, we remark that some related work on secure distributed computation and CAPTCHAs [9,10,2,1] has appeared in cryptographic literature. We are well aware that "security" is less of a cut-and-dried issue in the human computation context than in the cryptographic context, but we view this as an interesting research challenge. Of course, DHC also has interesting implications for algorithm & programming language design, and human-computer interaction.

4 Early Thoughts

We have used basic tools from probability theory and decision theory in the design and analysis of secure DHC systems. First, our analysis shows, interestingly, that in the presence of certain types of adversaries, standard tools like Bayesian inference are worse than simple approaches like majority vote for combining individual answers. Next, by trying to model candidate utility functions for end-users, we find several design principles: we should provide payouts to

[1] Server trust can be minimized by augmenting a DHC system with a voter and results-verifiable voting protocol [4].

clients in direct proportion to a rating that measures the accuracy with which they provide answers; we should decrease the rating substantially if a provided answer seems to be incorrect and increase it only slowly for answers that appear correct; and finally, we should take extra measures if a client's payout from cheating is potentially high.

We took this position previously [6] and subsequently discussed these issues in greater detail in [7]. While this work is still preliminary, it seems that secure *human* computing presents a new paradigm that is likely to suggest a rich set of research problems.

Acknowledgments

We thank Naresh Apte, Frank Bossen, Neil Daswani, Minoru Etoh, Ravi Jain, Khosrow Lashkari, Phil Mackenzie, and David Molnar, for helpful discussions and for pointing us to various examples of DHC.

References

1. von Ahn, L., Blum, M., Langford, J.: Telling humans and computers apart automatically. Communications of the ACM 47(2), 56–60 (2004)
2. von Ahn, L., Blum, M., Hopper, N., Langford, J.: CAPTCHA: Using hard AI problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656. Springer, Heidelberg (2003)
3. von Ahn, L., Dabbish, L.: Labeling Images with a Computer Game. In: ACM CHI 2004 (2004), http://www.espgame.org/
4. Cramer, R., Gennaro, R., Schoenmakers, B.: A Secure and Optimally Efficient Multi-Authority Election Scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
5. Drèze, X., Hussherr, F.: Internet Advertising: Is Anybody Watching? Journal of Interactive Marketing 17(4), 8–23 (2003)
6. Gentry, C., Ramzan, Z., Stubblebine, S.: Secure Distributed Human Computation (Short Form). In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 328–332. Springer, Heidelberg (2005)
7. Gentry, C., Ramzan, Z., Stubblebine, S.: Secure Distributed Human Computation. In: Proc. ACM Conference on Electronic Commerce (2005)
8. Goldreich, O.: Foundations of Cryptography, vol. 2. Cambridge University Press, Cambridge (2004)
9. Golle, P., Mironov, I.: Uncheatable Distributed Computations. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, p. 425. Springer, Heidelberg (2001)
10. Golle, P., Stubblebine, S.: Distributed computing with payout: task assignment for financial- and strong- security. In: Financial Cryptography 2001 (2001)
11. Prahalad, C.K., Hart, S.: The Fortune at the Bottom of the Pyramid. Strategy + Business (26), Q1 (2000)
12. Spam Net Web Site, http://www.cloudmark.com
13. Vipul's Razor Web Site, http://sourceforge.net/projects/razor
14. Wikipedia. Click Fraud, http://en.wikipedia.org/wiki/Click_fraud
15. Zhou, F., Zhuang, L., Zhao, B., Huang, L., Joseph, A.D., Kubiatowicz, J.: Approximate Object Location and Spam Filtering. In: ACM Middleware 2003 (2003)

Secure Distributed Human Computation
(Transcript of Discussion)

Craig Gentry

Stanford University

The premise is that humans can solve certain problems that computers can't, at least for the time being. There are basically two main applications of this idea: the talks on either side of mine are talking about using this premise as an automated Turing test, what I'll be talking about is a little bit different, using it as a motivation for actually harnessing human intelligence to solve problems that computers can't solve.

Here's a very simple example: some people don't have bank accounts or credit cards, but they have cheques that they want to cash, and you'd like to do this in an automatic way without actually involving a teller. So the question is, how do you authenticate this person? You have to check that the person that the cheque is made out to is the person who's trying to cash the cheque. AI hasn't made as much progress on the problems of facial recognition and biometrics that it would like to, so the very simple solution is just to use a human at the backend, and that's what this PayCash system does, they just take a photograph of the person, and they send it to a backend operator who confirms that it's the right person.

George Danezis: The problem is that you need to somehow make sure that the person cashing the cheque was the person to whom the cheque was made, how does biometrics help you, the cheque doesn't actually contain the biometrics of the person whose image you capture?

Reply: Well the person on the backend will have a face in its database to match the face of the person against.

Frank Stajano: So how do you get the signature in the first place?

Reply: They'll ask for your registration face.

Frank Stajano: So people without bank accounts or credit cards still have registered to the extent that their photograph is on-line in the bank's database.

George Danezis: In America they have a driver's licence.

Alex Shafarenko: I still don't quite understand because, at least in the UK, I write you a cheque, and this is my cheque, this comes from my chequebook, it doesn't contain your picture or your name, I just write your name down with a pen, and then I give it to you. The only thing binding this cheque to you is the name. How do you do then make sure that the biometric matches this name, because you've registered using the same name before with your biometric? What if the name is John Smith?

Reply: Oh, yes, but that's the same problem that you would have with a driver's licence.

OK, so a second problem is the image labelling problem.

Mike Bond: I would think though that the ESP game would only be fun if it was an interesting image; if it's a tile from the background of a webpage then boring is going to be the word.

Frank Stajano: And it would match with the other guy's description, and then they'll be onto the next image.

Reply: Well it's sort of recursive in the sense that the labels that you would get for a boring image would be things like, boring, or they wouldn't be able to agree on a common label, or something like that, so at that point they can do some filtering.

Tyler Moore: It's certainly hard work for these guys to produce the image set, they probably had to, to make it fun they'd probably have to spend some time and thought making a fun image set.

Reply: To some extent I think they did that, but they can also use the input that they get to do some filtering as well.

Matt Blaze: Did they actually do this, or is it simply a game that drives the images right now?

Reply: They actually do that, yes.

Matt Blaze: But if there's ESP involved why do you need a computer?

Reply: Well, that's a good point, I don't know how to respond to that one.

Sandy Clark: This is training, that's why they're playing it, they're practising.

Ross Anderson: You then maybe have a problem such as evolution where you have clustering between two outcomes, each of which is disputed by the other side, there's going to have to be some human intervention isn't there?

Reply: Yes, this is, I was going to mention this a little later, but it's to some extent an example of Arrow's Theorem, that there's no perfect voting protocol that's both fair and consistent, so the question is, how do you assign, how do you aggregate the input of different users in sort of a voting type of way? You could have a flat scheme where it's basically all majority vote, or you could have a system that's sort of based on the individual ratings of the users, and it's really unclear what you want to do in terms of malicious users, because different voting schemes give different vote results.

Ross Anderson: I would suspect that in a real system, what you would not do is to try and automate it all, because Arrow's Theorem basically works because circular preferences exist, and if I were implementing such a system for real, I would arrange it in whatever circular preferences emerged that an alarm would go off on the bridge and the human would pay attention to it. Similarly, if you get strongly bi-modal responses to one question, then that would suggest that somebody ought to take a look at what's going on.

Reply: But even for a single question you can have circular responses, can't you.

Ross Anderson: If you allow three or more preferences.

Bruce Christianson: There are also questions, for example involving the birthday paradox, where you'll get a very clear majority who will favour the wrong answer.

Reply: Yes, and this is, I think, an inherent problem of human computation. You are given the input from the humans, and there's no external thing that you can anchor it to[1], you just have to make do with what you get.

Ross Anderson: On a related point it might be worth thinking about whether you want a reputation system to be run by machines or by humans. If the computation that you're trying to break down was to write an operating system to compete with Windows, for example, then any practical way of organising ten thousand volunteers into a group that might actually get there, would involve people as individuals recognising who are the gurus amongst them.

My point is that people can pick gurus, but computers can't; this is perhaps another one for your list of CAPTCHA complete problems.

George Danezis: Traditional Bayesian models do not assume that there is a model for an adversary, so they try to look at whether something is a good or bad answer, and try to infer how reliable someone is, but do not know if they're an adversary or not.

Reply: Yes, that is actually a point, if you were to include in that Bayesian consideration, is this person a lurking adversary.

George Danezis: Yes, so if you have a model for someone who can be reliable or unreliable, but also an adversary or not, then I think you may actually find models that could fit that.

[1] Compare LNCS 1361, pp 105-113.

Reply: Yes, that's true, but that seems like a difficult problem.

Ross Anderson: On Wikipedia you can detect where there is contention; a simple velocity check would tell you (even if you didn't know it a priori), that George Bush's webpage would be one which the moderator should keep an eye on.

Tyler Moore: To go back a bit, I was surprised that you said that people in an on-line environment would be willing to accept less payment than in an in-house type scheme, because it would seem to me that it would be the opposite. You need to pay someone more in an on-line environment because you have no means of monitoring them, and controlling them, whereas if you have an employee, it's easier to monitor and so you have a greater assurance. . .

Ross Anderson: I think maybe you could externalise the risks of that. If outsourced employees are less valuable because they're less monitorable, then if there are fewer people buying the labour than selling it, the price is lower.

Mike Bond: I think it's an ability thing. I think for on-line people, we're getting their slack time when they're not doing the job that they're really paid enough to do, in-house people are just good enough to be able to label these images, that's why the labour is cheaper.

Bruce Christianson: But if they can get paid more for doing this at home, then they'll do that. If they can get paid more for doing it as an outworker they'll leave.

Mike Bond: Yes, but they may not be able to afford to.

Ross Anderson: A lot of people make a living from on-line games by grooming characters; start a rock character and working it up to level whatever, and then sell it on Ebay to someone for real cash. The people who do that tend to be teenagers in places like Romania, because if the most you can get for working in Tesco's is for £4 a week, if you can make characters for £20 a week it's a better deal.

Mike Bond: But I think there's an interesting point that carries on from here. You were saying, how do you build a right community of people to solve one of these big distributed human computation problems, and I believe it's not just about paying people, it's about creating the right incentives for a whole community, so it's not just paying people with loyalty points, or with lottery tickets, but creating a community where we all put up with our jobs. People put up with bad jobs because they're still part of the community of people, so it's not just a website that provides CAPTCHAs, but it's a website that provides chat, so that you can have a good laugh with your mates if you find a really funny image of somebody that you've got to ascribe a keyword to, build a whole community out of it. And then people put up with grime for the good aspects, and people build, the positivity comes out, and you find a way to solve this horrible problem while still having fun.

Ross Anderson: What you're trying to do here is to build a system that does management. Now the late Roger Needham had a very interesting take on this, he said, "administration is that for which an algorithm exists, everything else is management", and to make this run I think you have to have some of these management-type things, you know, annual productivity competition with a ticket for two to Barbados for the winner, tons of stuff like that. Stuff like that is cheaper as a means of motivating people than paying them more salary. Once people look at a basic sufficiency, they won't work so hard for extra cash, but they will work hard to get one of the fancy parking spaces in front of the building, or to get a company car with a door that closes with a satisfying clunk. Now what you've got to do is find the on-line equivalent of the satisfying clunk.

Reply: Well to some extent I think the ESP gamers know pretty well that they have a game which people actually perceive as being fun, they're just labelling images, but it's packaged in a way that is appealing to a lot of people I guess.

Steven Murdoch: I did some work on finding jpeg thumbnails. When you take a photo with a digital camera, it will make a thumbnail of the image, let's say in a jpeg, if you then alter the image, the original thumbnail remains unchanged, so I put a trawl which looked for lots of images which had a jpeg thumbnail which was different from the original. In a small number of cases removing something embarrassing, and in a large number cases it was just completely boring, where someone had cropped out when they took a photo with too much sky. So Max Dornseif had a website where he showed the original image and the thumbnail, and we took a vote on how interesting it was. We found a lot more interesting images that way[2].

Reply: Yes, and people probably enjoyed that, perversely.

Mike Bond: But that's really bad grind; this is an example of the lengths people are prepared to go to to do something even more boring than ESP, if there is a fun reward.

Frank Stajano: So what embarrassing things can you do just by cropping, for example?

Steven Murdoch: Well, someone took a photo of themselves naked and then removed the parts that showed any evidence of that. Then she published it, because it was a carefully innocent photograph.

[2] See `http://md.hudora.de/presentations/#hiddendata-21c3`

Reply: CAPTCHAs are sort of an artificial problem, right, but DHC problems typically are real world problems, and it's not clear at all how a computer is going to generate automatically an instance of a DHC problem of interest.

Ross Anderson: I suppose it depends on your assumptions about what proportion of your user population are hostile. If you're Google, and more than five percent of your users are Yahoo, if somebody other than Yahoo attacks you, they're going to be less than five percent of your users. So rather than using ringers you just use voting, and compare people's responses with each other.

Reply: Yes, if you don't even need ringers that's much easier.

Sandy Clark: My problem with voting though is you run into the issue of one particularly motivated group, for example, the American religious conservatives. If you're using majority voting are you going to have this one politically or religiously motivated group deciding on something?

Reply: Yes, this is more of a collusion question.

George Danezis: But, sorry, it also comes back to this idea that for most problems that humans are very good at, you require humans because actually there is no objective truth, like the Wikipedia entries on contentious issues. It isn't the case that there is some truth, and if you get more, and more opinions, you will approximate it, it's just that people are quite happy to fight over Wikipedia entries until one of the two becomes no more. So here is an inherent complexity that you can't factor out.

Ross Anderson: Well one approach to this would be to say, fine, we're not going to say right or wrong, we're just going to say, custard. So you get the Republican page on George Bush, and you get the Democrat page on George Bush, where's the problem.

Reply: Yes, that's interesting if you can do it, not only just acquire, say, labels for images, but sort them in some sense by the distribution of people that they come from; there might be a bi-modal distribution for the descriptive terms, so you might want to separate those out. That's a good point.

James Heather: Wikipedia is not a particularly compelling example because the ethos of Wikipedia is that there is an objective truth, and that's what they're trying to get at. That's why you're not supposed to put up opinions, because you're supposed to put up purely factual stuff. The disputes come over whether something is factual or opinion.

Reply: OK, so how do we prevent collusion? To some extent you can solve this by randomising problem assignment, so for the religious groups the result of randomly assigning the problems is that hopefully they would get lots of problems in which they have no special interest.

Tyler Moore: One other potential problem with that is, if you randomly assign problems that people can solve, they might get adventurous; they might instead prefer to choose what problems they want to solve. In Wikipedia, for instance, people want to write about what they know and are interested in, not random essays.

Reply: Yes, that's a good a point, the ESP game actually has theme rooms, so if you have a special interest in something then you can do that. I guess that causes a problem in terms of random problem assignment.

Ross Anderson: What that comes to is that you will, I suppose, end up having to recreate many of the other functions of a modern large corporation, you have to have a human resources department, and have to have training departments, you have to figure out how you match your volunteers to the problems, you have to have analysis departments so that you can understand an application and figure out, whether you're going to have to use clustering, or account for the modality of worker preferences, and so on. So once one starts to think about real applications, this isn't a thing that you can do purely with computational tools, you're going to need a very much richer structure in order to make it work in the real world.

Reply: Yes, I guess the hope would be that the amount of output produced more than offsets the cost of the core management team.

Mike Bond: So are you trying to say the software becomes more and more generic, eventually the software core becomes Microsoft Office.

Ross Anderson: No, the software core ends up looking pretty much like Oracle Corporate Financials. You may end up with slightly more palatable individuals in the various corporate roles tackling some particular problem, but for many large problems you could end up having many of the same functions having to be discharged, and many of them couldn't realistically be done just by computer, you need human involvement, so how you organise that mix of human and machine becomes the interesting problem. So in effect what we're trying to do is to produce a more generalised and more palatable version of what companies in India do.

Reply: Yes, this is basically a form of human employment, so basically the trick is to try and get people to do stuff, basically for free, so a form of slave labour, I guess.

Mike Bond: I think that the generality that Ross presents is very difficult to solve, and the big boys tried to do that, and it's not very good anyway. But for writing software, or for image overwriting we have various examples where that works well. So maybe we should concentrate on what is so special about these, an assertative management that is inherent in a specialist structure, or whatever?

Ross Anderson: You know, I reckon the last chunk of this is trust. One reason that poor countries are poor, is that the poor get ripped off by the richest, and so they've got no incentive to make any effort. Even in Scotland until the 19th century in the Highlands, as soon as any agricultural improvement came along, and the peasants had more cows and more sheep, the landlord's factor would come along and put up the rents, and so the peasants found it didn't matter how hard they worked, they always ended up at the subsistence level, while the Duke of Argyle got himself fancier and fancier castles. Now we fixed that by means of social mechanisms, a whole raft of them, which in turn generates trust. We've got labour laws, landlord and tenant laws, and so on: that's missing in places like rural India. If you're going to use this as a structure to get people in rural India with a primary school level education to do useful work, one of the main things that you can deliver to them is the fact that their wages will actually be paid, and they'd be paid in full in hard cash, rather than in money that can only be spent in the company store. So what you may be providing that's valuable, if you make this into a general framework, is not something technical, but something that is fundamentally societal, namely trust, in an employment context. And actually, if you give somebody points in an on-line game, that is in some sense a lot harder for the local government in a third world country to tax, than if you hand them cash rupees, so perhaps you have to rejig the financial system, in interesting ways.

Reply: Yes, I think it levels the playing field to some extent.

George Danezis: So what you are saying is, if you can connect people to a remote economy which is under someone else's control, then you can help those people and exploit them too.

Ross Anderson: Well the point is, you're exploiting them less than the local cane merchant; you're stealing the local cane merchant's peasants. In a medium term this will affect the action of the local cane merchant.

Sandy Clark: But it's an underground economy.

Bruce Christianson: That's part of why they have the incentive to do things for you.

Mike Bond: Until the age of the Internet we've never had the opportunity to have remote economies, that's the difference.

Reply: There are also some problems where you can't really expect to be able to assign the problems randomly, for example, spam, I mean, users just get the spam that they get, and you can't have a situation where their spam is forwarded to someone else, that wouldn't work.

Ross Anderson: Why not? Why can't I hire myself a secretary in Bangalore for £2 a day to read all my email and throw away the spam?

Reply: Usually you'd want to build up some trust with a secretary.

Bruce Christianson: But if you're farming, this is precisely the problem that this is solving. Farming the spam out to lots of people, and go for the voting protocol is this spam or not.

Reply: Yes, you might, you could do that, but you might be afraid that you would receive sensitive email, so using this as a global system is somewhat dangerous. But you could have individual cases where users are willing to do that.

Bot, Cyborg and Automated Turing Test
(Or “Putting the Humanoid in the Protocol”)

Jeff Yan

School of Computing Science
University of Newcastle upon Tyne, UK
Jeff.Yan@ncl.ac.uk

Abstract. The Automated Turing test (ATT) is almost a standard security technique for addressing the threat of undesirable or malicious bot programs. In this paper, we motivate an interesting adversary model, cyborgs, which are either humans assisted by bots or bots assisted by humans. Since there is always a human behind these bots, or a human can always be available on demand, ATT fails to differentiate such cyborgs from humans. The notion of “telling humans and cyborgs apart” is novel, and it can be of practical relevance in network security. Although it is a challenging task, we have had some success in telling cyborgs and humans apart automatically.

1 Introduction

Nowadays, CAPTCHA (Completely Automated Turing Test to Tell Computers and Humans Apart) [2] is almost a standard security mechanism for addressing undesirable or malicious bot programs such as:

– Voting bots, which could cast thousands of votes as masquerading humans in online polls [2];
– Email account registration bots, which could sign up for thousands of accounts every minute with free email service providers [2];
– Email spam bots, which could automatically send out thousands of spam messages every minute [2];
– Weblog bots, which could post comments in weblogs pointing both readers and search engines to irrelevant sites; and
– Search engine bots, which could automatically register web pages to raise their rankings in a search engine.

The basic idea of CAPTCHA is to force there to be a human in the loop – it works as a simple two-round authentication protocol as follows.

S(ervice) → C(lient): a CAPTCHA *challenge*
C → S: *response*

A CAPTCHA *challenge* is a test that most humans can pass but current computer programs cannot pass. Such a test is often based on a hard, open problem in AI, *e.g.* automatic recognition of distorted text, or of human speech against a noisy background. Differing from the original Turing test [1], CAPTCHA challenges are automatically generated and graded by a computer. Since only humans are able to return a sensible *response*, an automated Turing test embedded in the above protocol can verify whether there is a human behind the challenged computer.

Although the original Turing test was designed as a measure of progress for AI, CAPTCHA is a security mechanism. It is straightforward to apply CAPTCHA to prevent a bot say from distributing spam emails: an email is not forwarded until it is confirmed using a CAPTCHA challenge that there is a human behind the computer sending the message.

However, in this paper, we show that CAPTCHA is not a panacea for dealing with the threat of undesirable or malicious bots. We motivate an interesting adversary model, cyborgs. A cyborg, according to the *Oxford English Dictionary*, is a creature that is part human and part machine. In reality, it can be either a human assisted by bots, or a bot assisted by humans. Since there is always a human behind the bot(s), or a human can always appear behind the bot(s) on demand, CAPTCHA will fail to differentiate such cyborgs from humans. We also show the novel notion of "telling humans and cyborgs apart" can be of practical relevance in network security.

The rest of this paper is organised as follows. Section 2 looks into two popular bots and shows why CAPTCHA is doomed to fail to deal with such bots. Section 3 presents a simple model explaining when CAPTCHA works in dealing with bots, and when it does not. Section 4 briefly introduces some of our work in automatically telling humans and cyborgs apart, and Section 5 concludes.

2 CAPTCHA: Not a Panacea!

In this section, we use two popular cheating bots in online multiplayer games to show that CAPTCHA is not a panacea for dealing with the threat of undesirable or malicious bots.

2.1 MMORPG Bots

MMORPG stands for "Massive Multiplayer Online Role-Playing Game". This genre of games runs an evolving virtual world on a server. Thousands of human players can play on the server over the Internet at the same time, each playing the role of a virtual character, which can for example be a medieval knight or an ancient Chinese kung-fu warrior depending on the game settings.

An MMORPG bot is a program that automatically plays the game on behalf of a human player. Such a program is usually game-specific, and driven by predefined scripts. MMORPG bots can be easily configured to perform many activities that a human player would do in the game, such as move, kill, cast spells and collect items. Some such bots also allow a human player to input instructions to directly control his character at any time. That is, MMORPG bots can be both script-driven and interactive.

With the help of MMORPG bots, which never get tired, cheaters can reap rewards with much less effort than their honest counterparts. The use of such bots is usually forbidden by game operators. The common practice for MMORPG game operators fighting against bots relies on human policing: a game master patrols game zones, recognises and questions suspicious players. Employing CAPTCHA would appear to be a good solution for keeping bots out of MMORPGs at a first glance, since MMORPG bots are in fact usually left unattended once they are connected to a game server.

Before attempting to reach any conclusion regarding the use of CAPTCHA in MMORPGs, we perform a simple security analysis as follows. We make the standard assumptions that the game server is under the control of a game operator, and secured by standard technologies, and that a game client can be under total control of a cheater (since it is run on his own computer, which is hard or expensive to be made tamper-proof). We also assume that the CAPTCHA facility is properly implemented by qualified security engineers, having addressed the following security issues.

– All critical things such as randomness used for generating CAPTCHA challenges are implemented in the game server. In addition to supporting game play, a client merely displays CAPTCHA challenges, accepts responses from a player, and returns them to the server for verification.
– An adversary should not be able to gain, by intercepting network traffic, anything that helps solve a CAPTCHA challenge. A naive mistake made by one student of mine was as follows. Each random string used for CAPTCHA challenges was generated on the server side, and it was transmitted in plaintext to a client. The client then distorted the string, embedded the distorted text, together with some noise information, into an image, and finally displayed the image. This implementation would allow a bot to easily pick up the right answers from the traffic! Encryption would not help, since the client is assumed to know everything. Rather, each CAPTCHA challenge should be prepared by the server, and then transmitted in an image format across the network to the client.
– Dictionary based replay attack. Adversaries should not be able to benefit by collecting old challenges and their answers into a dictionary, and then replaying old correct answers.

Unfortunately, such a CAPTCHA based bot defence is still vulnerable to the following attacks:

– **Man in the middle (MITM) attack.** This is not new. It was alleged that a spammer could make use of the MITM attack, shifting the load of solving CATPCHA challenges to porn site visitors [6]. This kind of attack is certainly doable, and could be exploited by MMORPG bot users.
– **Outsourcing attack.** Bot users can outsource both their game play and the task of solving CAPTCHA challenges to people in low-paying countries.
– **Housewife attack.** A key observation is that it is often possible for a bot to differentiate each CAPTCHA challenge from other game events. Thus,

a human can be alerted by the bot to answer the challenge in time. Bot attending in MMORPGs could then become an attractive profession for housewives, who would make money by attending their bots occasionally (*i.e.*, upon each alert), while looking after their household business as usual.
- **Collusion attack.** Bots can be made to communicate with each other, and then each alert can be propagated across the bot network. Therefore, a CAPTCHA challenge can be attended by either a cheater or one of his friends, whoever is available.

There is also a usability concern (beyond accessibility) when integrating CAPTCHA into MMORPG games: you cannot issue CAPTCHA challenges very often unless you want to ruin the fun of game play! Such a concern makes it even easier for all the above attacks to succeed.

2.2 Aiming Robots

Aiming robots (or aimbots for short) have frequently been used as a cheating tool in online multiplayer shooting games, *i.e.*, first-person shooter (FPS) games. An aimbot works as either a client hook or a proxy sitting between a client and a game server. In both cases, the bot tracks the movements and locations of players. Whenever a cheater issues a Fire command, his aimbot could automatically pick a target, and point the cheater's weapon straight at the selected target. An advanced aimbot can also pretend to act like an ordinary human being, either by switching itself on and off periodically, or by intentionally missing targets from time to time.

CAPTCHA does *not* appear to be applicable to prevent aimbots, since

- First, a human player is always behind his aimbot. Such a bot largely does aiming (as well as target tracking) only, and it cannot perform other in-game tasks on behalf of a player, which usually require human involvement due to characteristics of such games. Thus, the player could always be available to respond to CAPTCHA challenges, if any.
- Second, FPS games are usually fast paced. A CAPTCHA challenge would be too disruptive for game play. Due to this usability concern, few game designers and players, if any, would consider CAPTCHA as an acceptable solution. (Such a usability concern is not so serious in MMORPGs, which are usually slower paced.)

3 A Simple Explanation

Apparently, the following three notions are different:

- "Breaking CAPTCHA"

  "Breaking CAPTCHA" usually means "breaking CAPTCHA challenges", and it is as hard as solving the underlying AI problem.

– "Breaking CAPTCHA based authentication"

  "Breaking CAPTCHA based authentication" is to fail the CAPTCHA protocol's authentication purpose. Surely, either "breaking CAPTCHA" or the MITM attack will lead to such a protocol-level failure, alone. But the MITM attack does not need to solve the underlying AI problem.

– "Defeating CAPTCHA based bot defence"

  It is true that unless the underlying AI problem is solved, CAPTCHA challenges cannot be answered without having forced there to be a human behind the challenged computer. This, however, does not guarantee that the CAPTCHA mechanism is a good defence against all bots. First of all, there are circumstances in which arranging such human involvement is quite feasible. Thus there is the possibility that all CAPTCHA challenges to a bot could be properly answered, so bots would not be stopped – instead, they would remain undetected.

  On the other hand, as the earlier discussion of the MITM attack, the outsourcing attack and the collusion attack have revealed, the CAPTCHA mechanism could ensure that there was a human behind a computer, but it might not be the "right" human!

  Moreover, usability concerns can also render the CAPTCHA mechanism less effective as expected or even useless in defending against some bots.

The CAPTCHA mechanism can be refined to some extend to address the above "not the right human" problem. For example, when challenges that demand a *specialised skill* or a certain *cognitive ability* to solve are used, they will force there to be a human with the required skill or capability in the loop. For instance, to use "life and death" puzzles in Go games as CAPTCHA challenges can provide a guarantee that people behind a computer are humans who understand Go. In addition, a biometric signature such as keystroke dynamics might be embedded in one's response to a CAPTCHA challenge. All this might be sometimes sufficient for authentication purposes. But none can be a generic way of defeating undesirable or malicious bots due to the obvious disadvantages.

To explain when the CAPTCHA mechanism works to defeat bots and when it does not, we define a simple function as follows to describe the cost, denoted by C, that an adversary has to bear to beat such a bot defence.

$C = f \times t \times c$
f: The frequency of CAPTCHA challenges, *i.e.*, the number of challenges occurring per unit time.
t: Time period during which CAPTCHA challenges will occur.
c: Average cost for an adversary to solve a single CAPTCHA challenge.

The number of challenges an adversary has to solve in the given t time is defined by $f \times t$. To simplify our discussion, we assume that t is constant.

Thus, the CAPTCHA mechanism is effective at defeating bots when $f \times c$ is prohibitively high. The authors of [2] seem to have implicitly assumed that this condition always holds in various bot defence scenarios, but this assumption is not necessarily true. Apparently, all the bots listed in Section 1, as well as others not discussed in this paper, are similar to MMORPG bots in that a CAPTCHA based defence will be vulnerable to all four attacks discussed in Section 2.1.

The condition $C = 0$ or $C \approx 0$ depicts scenarios where the CAPTCHA mechanism fails to prevent bot attacks. Such scenarios include the following cases.

- When the hard AI problem exploited for CAPTCHA tests becomes tractable by a program, $c = 0$ and thus $C = 0$. (We ignore the cost that an adversary has to bear for writing such a program. For example, the program might be downloaded from the Internet, free of charge.)
- When the adversary can make use of the MITM attacks, *e.g.*, shifting the load of solving CATPCHA challenges to porn site visitors, $c = 0$ and thus $C = 0$. (Similarly, we assume that the cost of implementing such MITM attacks are negligible.)
- In the aimbot case, $f = 0$ or $f \approx 0$ (otherwise it would be disruptive for game play), and thus C is negligible.

When f is large and $c \neq 0$, an adversary can defeat the CAPTCHA mechanism by lowering his average cost c, *e.g.* by outsourcing the task of solving each challenge.

In the MMORPG bot case, f must be small, and c can be lowered by the adversary through either the outsourcing attack or the housewife attack. That is, an adversary can also work around the CAPTCHA based bot defence.

4 Tell Humans and Cyborgs Apart

An aimbot, together with the human player using the bot, is a good example of cyborg. So is the combination of an MMORPG bot and a human behind it (a housewife, a porn site visitor or an outsourcing worker). Since there is always a human behind the bots, or a human can always appear behind them on demand, the CAPTCHA mechanism cannot tell humans and such cyborgs apart at all.

People might wonder: given that an adversary has to arrange that his bot is attended, why does he bother to use it in the first place? This can be simply explained as follows.

- First, bots can be superior to ordinary human beings in some aspects. For example, an aimbot can achieve better aiming accuracy; an MMORPG bot can be more patient at repeating tedious tasks. Thus, while bots do things that they are good at, the adversary just has to attend CAPTCHA challenges, and probably some other easy tasks as well. This is a good analogy of "Render unto Caesar what is Caesar's and render to God what is God's".
- Furthermore, while f or c is reduced due to usability concerns in some scenarios, the amount of human labour required to attend CAPTCHA challenges will be reduced, making the adversary's life even easier!

To tell humans and cyborgs apart automatically is a real security concern in online games, one of the most popular Internet applications, as well as in other contexts. For example, with the help of an aimbot, a cheater can easily achieve a high aiming accuracy and beat his opponents, gaining himself an unfair advantage over honest players. A cheater can also run bots in MMORPG games, collecting virtual items without real play and then selling them, for example on eBay, for real money – thus achieving an unfair financial gain. Botting (*i.e.* running bots) violates fair play, an important security concern in such settings [3,4]. Now that first-person shooters are emerging as spectator-games[1], fairness enforcement becomes an even more serious issue in these competitive games. In addition, botting in MMORPGs can also undermine the delicate balance of the game world, a critical factor affecting the success of such games.

Although the task is challenging, we have had some success in automatically telling cyborgs and humans apart. In the following, we briefly discuss our work on differentiating honest game players from cheaters assisted with bots (*i.e.* cyborgs). Our discussion is focused on the aimbot case, but our method, in principle, can be applied to identify MMORPG bots as well.

4.1 Aimbot Detection

We have developed a Dynamic Bayesian Network (DBN) based statistical inference approach to distinguish honest players from aimbot cheaters in FPS games. In our DBN model [5], the probability distribution of a player's aiming accuracy is dependent on the following random variables:

– Whether the player is cheating or not, denoted by the random variable $\mathcal{C}$;
– Whether the player is moving or not;
– Whether the aimed target is moving or not;
– Whether the player is changing the aiming direction;
– The distance between the player and the aimed target.

By learning the conditional probabilities from a set of training data, our algorithm can infer the probability of the missing variable $\mathcal{C}$ in other sets of data. In our initial experiments, this algorithm has differentiated all aimbot cheaters from honest players with no false positives.

Our ongoing work is to refine our algorithm so that we can distinguish between aimbot cheaters and professional players, who may achieve an outstanding aim accuracy in most cases. We rely on a simple intuitive observation: while being able to achieve a high aiming accuracy, a super human player is (by definition) also good at other aspects of the game play. But this is not necessarily the case with aimbot cheaters. (Indeed there might be some all-around aimbot cheaters who perform well in all these aspects, not much could be done to detect them automatically however.)

[1] Like traditional sports such as football and basketball, FPS tournaments now attract lots of spectators. Top professional game players (or the so-called "cyber athletes") can make a lucrative living with commercial sponsorships just like traditional sports stars.

So given that super players and aimbot cheaters both pass the threshold used in our previous DBN model [5] to identify cheaters, an additional correlation test will help: if there is a positive correlation between a player's high aiming accuracy and his other "performance factors" such as movement agility, mistake counts/frequencies, appropriateness of weapon choices and efficiency of ammunition use, we are more comfortable (than before) assuming that he is a super human player; otherwise we are more confident that we are dealing with a cyborg, *i.e.*, the suspect is an aimbot cheater.

5 Conclusion

CAPTCHA is not a panacea in dealing with the threat of undesirable or malicious bots. Often, it is insufficient to simply verify that there is a human behind a computer. Usability concerns beyond accessibility can also render CAPTCHA inappropriate for some application contexts. Instead, "telling humans and cyborgs apart automatically" can be a relevant and generic security problem, to which CAPTCHA fails to provide a solution. However, it appears to be promising to automatically differentiate cyborgs from humans by recognising the characteristics of both the machine and human facets of cyborgs.

Acknowledgement

The author thanks Brian Randell for valuable discussions and comments, Peter Ryan for suggesting this paper's subtitle, and Richard Clayton for pointing us to his investigation of the authenticity of the MITM attack story in [6] at this workshop.

References

1. Turing, A.M.: Computer machinery and intelligence. Mind 59(236), 433–460 (1950)
2. von Ahn, L., Blum, M., Langford, J.: Telling Humans and Computers Apart Automatically. Communications of the ACM 47(2) (February 2004)
3. Yan, J.: Security Design in Online Games. In: Proc. of the 19th Annual Computer Security Applications Conference (ACSAC 2003), Las Vegas, U.S.A, December 2003. IEEE Computer Society, Los Alamitos (2003)
4. Yan, J., Randell, B.: A Systematic Classification of Cheating in Online Games. In: Proc. of the 4th ACM SIGCOMM Workshop on Network & System Support for Games (NetGames 2005), October 10-11. ACM Press, New York (2005)
5. Fung, Y.S., Lui, J., Liu, J., Yan, J.: Detecting Cheaters for Multiplayer Games: Theory, Design and Implementation. In: The second IEEE International Workshop on Networking Issues in Multimedia Entertainment, Las Vegas, USA (January 2006)
6. Clayton, R.: Automating responses to email challenges (February 2006), http://www.cl.cam.ac.uk/~rnc1/cr/index.html

Bot, Cyborg and Automated Turing Test

(Transcript of Discussion)

Jeff Yan

University of Newcastle upon Tyne

Ross Anderson: Bot tending might be an attractive activity for children, because children could receive the challenges on their mobile phones, to which they are almost physiologically attached these days, and they're perhaps used to relatively smaller amounts of pocket money.

Mike Bond: You talked about routes for sending CAPTCHAs which go outside the game; given that the bot has control of the client, what about sending the CAPTCHA back into the game to a human player who is maybe indifferent about bots, and then paying him a virtual currency to solve it? The client would have both the infrastructure to reinsert the CAPTCHA, and to make a payment, there and then.

Reply: Bots can communicate with each other, right, so you can implement this in many different ways, either inside the game or outside the game.

Ross Anderson: Another approach is to say that anyone can use an aimbot if they want, and you cluster competitive markets in aimbots. Some people play Baroque music on acoustic instruments, other people play music with sequencing backup tapes, and so on, the two do not compete. But it is quite possible for there to be massively online games where cheating is allowed.

Mike Bond: And there's a category of first person tactical shooters, where the game has been designed so it's much more important how you work in a team, and what your military tactics are. As long as everybody has a decent game, these sorts of things aren't a problem.

Ross Anderson: I suppose the problem is that if you have an orchestral competition for acoustic instruments, you can actually look and see that none of the violins have these electric pickups attached to them, but if you have a hundred people round the world playing a first person shooter game, you can't check for the absence of aimbots unless you do things like trusted computing, which brings other problems.

Reply: Yes, this is entirely server based solution, and it is scaleable, the CPU consumption is linear to the number of players, so there are two very good features in this solution.

Frank Stajano: Surely it should have the speed of the target in there somewhere?

Reply: Speed, well actually this is the first model we have used. And in our initial experiments we have got very encouraging results, there's no false positive, we can identify all cheating players from the honest players. We have a paper published in Networking Issues in Multimedia Entertainment, in collaboration with Yeung Sui Fung, a PhD student who works with me and my colleagues in Hong Kong. Lots of details you can find in this paper, but it is largely about the application of Dynamic Bayesian Network in detecting aimbot users.

Peter Ryan: Presumably we can do countermeasures and design bots that would get under the radar for such detectors?

Reply: Ah yes, it is possible, theoretically. For example, a secure multiparty computation can be applied to prevent this aimbot, but the problem is there is no efficient, practical secure multiparty computation protocols which we can use, especially this game implies everything is exchanged at a very fast pace.

James Malcolm: Isn't this a little bit like any other kind of performance enhancing drugs, in that if you're very much better than everybody in the world, then there's something suspicious going on, but if you're only just a tiny bit better then maybe you're just the best.

Reply: Well actually that's what I'm going to talk about. We rely on the following intuition: a very good human player is by definition also good at other aspects of the gameplay besides perfect at aiming accuracy, but the cheaters are not necessarily the case. So we just need to do a simple correlation test between aiming accuracy and other performance factors in the game, this test will say whether this guy is a real aimbot user, or a super player. Hopefully we can get some experimental results in the near future.

Chris Mitchell: What's your method that points to a noisy aimbot?

Reply: Well actually in our experiments we implemented three aimbots. The first one is a basic one, which it can achieve perfect accuracy all the time, and the second bot, which can switch itself on and off periodically, and the third one we have tried to intentionally miss some targets, so I think with these two advanced features, they belong to the new definition of noisy force, right. And in our experiment results no false positive, no false negative. So I think the power came from the Dynamic Bayesian inference techniques.

Frank Stajano: Have you tried this on any aimbots done by real people instead of done by you?

Reply: Not yet because we are using an opensource APS game, which you are welcome to have a try, it's cube, you can find it at www.cubeengine.com, which is written by a professor in the States.

Some students in Stanford wrote a very good aimbot, but the problems were they had not made the source code available. Another concern is that many bots are actually quite game specific because each game uses different network protocols, different implementations. We stick on this cube engine, is because it's opensource, and is quite well written, it's written in C++.

Mike Bond: I think probably there is a large gap at the moment between the way that publicly available aimbots move your view and the way that a human is physically capable of moving with a mouse. In particular humans move and refine, so if you think about how you use the mouse as rapidly as possible from one button to another, of which there's plenty of research on, you take a broad move and then you refine. The same is approximately true for aiming, so I image a simple aimbot must have a massive signature in terms of difference in the way that half of the aim goes towards the destination.

Reply: No, we're shooting in convergence.

Ross Anderson: That can be fixed. And that could be highly significant in the context of click fraud, which is the biggest headache that Google has at the moment, as far as advertising goes. You can be paying $10 a hit for a top rated search ad on Google, and if you tell Google, I will spend $10,000 this month, then as soon as a thousand people click, your ad drops off. So your competitor gets a bot to click a thousand times on your ad, so you go away. Or alternatively people have been placing ads on their own sites and then cheating by clicking on these ads so that they can get a share of the revenue. Now if you can use techniques such as tracking the pattern of the mouse movement that goes to the ad...

Frank Stajano: But that's invisible to the server.

Ross Anderson: Well Google may replace your desktop client.

Mike Bond: I'm not sure if mouse acts are considered privileged in Sandbox terms.

Steven Murdoch: If you change the cursor you can read what its locations are.

Mike Bond: So in theory you can run something on a client site?

Ross Anderson: This is the thing that above all else bugs Google at the moment because they're facing a number of class actions from disgruntled advertisers.

Reply: Yes, actually Peter Ryan recently invited a student to visit Newcastle who had worked with Google last summer doing a summer internship; he actually worked almost solving this problem, mentioned by Ross, exactly the same problem. As I was told, Google has got some solutions, but they have not published yet.

Mike Bond: Another solution that springs to mind is to change the game to artificially induce a maximum level of accuracy, which is below pinpoint accuracy. You know, as you breathe in and out your crosshair waves around, as it would in real life. And it occurs to me that for the cyberathelete club, it's important for the prestige of manufacturers funding this, that the tournament be seen to be fair, but for the levels below, people who are cheating this way are only upsetting people because it spoils the fun feeling like there's a cheat in the game. A superhuman player (who is not cheating) will spoil the fun as well, and if you can change the game to allow somebody to win, but still make you feel like it is a fair fight, like you have a chance of winning, that's a good start.

Reply: But the thought you have a chance to win the game from those superplayers, those professional players, they play the games very amazingly, unbelievable.

Mike Bond: I mean, it's not much fun running the hundred metres against an Olympic athlete, because you're almost certainly not going to win. Once it does get to that level, they will definitely beat you, and if you're in it to beat them, then you're not going to, unless you really have put some fantastic training in, or something terrible goes wrong with them.

Ross Anderson: I suppose the big difference between a first person shooter game and something like a click fraud is that in a shooter game if somebody's cheating you want to throw him off right now, whereas if Google discovers that they're getting fraudulent clicks from some source or another, they find out about the fraud twelve hours after the event, then they just send out smaller invoices at the end of the month. So perhaps something like click fraud gives you a more practical experimental sandpit, provided you can rope in one of the ISPs that actually run advertising.

George Danezis: Another solution for Google is to have a more volatile market; if some people start clicking like crazy on a particular word, then the value of that word decreases.

Ross Anderson: Well apparently if you click four times on a word from the same IP address, Google just sends off a reset, but how could you incorporate a CAPTCHA into web search, that really is a usability question.

Peter Ryan: Different topic, just out of curiosity, do the people visiting these porn sites ever ask themselves why they're being presented with a CAPTCHA?

Reply: I have no answer but this. But theoretically it is doable because you can find some implementations on the Internet.

Frank Stajano: Maybe they are good at making the ESP game interesting?

Reply: Actually in my paper, I have some discussions about the scenarios where a CAPTCHA will work, and scenarios where a CAPTCHA will not work, just a simple economic explanation.

A 2-Round Anonymous Veto Protocol

Feng Hao and Piotr Zieliński

Computer Laboratory, University of Cambridge, UK
{feng.hao,piotr.zielinski}@cl.cam.ac.uk

Abstract. The dining cryptographers network (or DC-net) is a seminal technique devised by Chaum to solve the dining cryptographers problem — namely, how to send a boolean-OR bit anonymously from a group of participants. In this paper, we investigate the weaknesses of DC-nets, study alternative methods and propose a new way to tackle this problem. Our protocol, Anonymous Veto Network (or AV-net), overcomes all the major limitations of DC-nets, including the complex key setup, message collisions and susceptibility to disruptions. While DC-nets are unconditionally secure, AV-nets are computationally secure under the Decision Diffie-Hellman (DDH) assumption. An AV-net is more efficient than other techniques based on the same public-key primitives. It requires only two rounds of broadcast and the least computational load and bandwidth usage per participant. Furthermore, it provides the strongest protection against collusion — only full collusion can breach the anonymity of message senders.

1 Introduction

Chaum introduced the *dining cryptographers problem* in 1988: three cryptographers want to find out whether NSA or one of them pays for the dinner, while respecting each other's right to make a payment anonymously [1].

In the same paper, Chaum provided a well-known solution: the dining cryptographers network (or DC-net). A DC-net uses "unconditional secrecy channels" to setup pairwise shared keys and an authenticated broadcast channel to send anonymous messages whose senders are untraceable. Details of DC-nets can be found in [1].

Despite their importance in anonymity research, DC-nets are not widely deployed for practical applications. The major problem is their requirement of *pairwise* shared keys. Setting up these keys relies on unconditionally secret channels [1]. The number of such channels grows squarely with the increasing network size, as does the total number of the shared keys. Message collisions are also problematic in DC-nets. If a collision occurs, a retransmission needs to be arranged. However, as we explain in Section 2.3, there are circumstances where retransmissions cannot resolve the collision problem. Finally, DC-nets are subject to various forms of disruptions [2]. Solutions to prevent disruptions make the system more complex.

We would like to highlight that the DC-net is not the only solution to the dining cryptographers problem. Essentially, DC-nets are designed to determine the boolean-OR of bits contributed by participants, while preserving the privacy of individual inputs [1]. Alternatively, the circuit evaluation technique can be applied to compute the boolean-OR function securely [18,11]. However, due to its generality, the circuit evaluation technique is expensive and impractical [10]. This will be explained in Section 3 in more detail.

We consider the dining cryptographers problem from a different perspective — suppose the three cryptographers vote against the statement: “no cryptographer has paid”. If anyone vetoes, it means “one of the cryptographers has paid”. Otherwise, it implies “NSA has paid”. Thus, an anonymous veto protocol can solve this problem well. Several such protocol designs exist [12,13,10].

In this paper, we propose a new veto protocol: anonymous veto network (or AV-net). Our solution is simple and very efficient. As opposed to DC-nets, AV-nets require no secrecy channels, have no message collisions, and are more resistant to disruptions. Compared to other veto protocols [12,13,10], AV-nets are more efficient in nearly every aspect, such as the number of rounds, computational load and bandwidth usage. In the rest of the paper, Section 2 explains the protocol and analyzes its security properties. Section 3 examines the efficiency of the protocol and compares with prior art.

2 Protocol

Our protocol does not require any private channels or third parties. It only assumes an *authenticated broadcast channel* available to every participant. In fact, this assumption is made in all past work in this line of research [1,11,12, 13,10] (see Section 3). It suffices to know that such a broadcast channel can be realized using physical means or digital signatures [1].

The protocol setting resembles the real-life situation quite closely — when people engage in public discussion, every word uttered can be traced to its originators. How can a participant with the veto right to say “no” *anonymously* in such an open environment? Our solution is provided below.

2.1 Two-Round Broadcast

Let G denote a finite cyclic group of prime order q in which the Decision Diffie-Hellman (DDH) problem is intractable [3]. Let g be the generator. There are n participants, and they all agree on (G, g). Each participant P_i selects a random value as the secret: $x_i \in_R \mathbb{Z}_q$.

Round 1. *Every participant P_i broadcasts g^{x_i} and a knowledge proof for x_i.*

When this round finishes, each participant computes

$$g^{y_i} = \prod_{j=1}^{i-1} g^{x_j} \Big/ \prod_{j=i+1}^{n} g^{x_j}$$

Table 1. A simple illustration of $\sum_{i=1}^{n} x_i y_i = 0$ for $n = 5$. The sum $\sum_{i=1}^{n} x_i \, (\sum_{j=1}^{i-1} x_j - \sum_{j=i+1}^{n} x_j)$ is the addition of all the cells, where +, – represent the sign. They cancel each other out.

	x_1	x_2	x_3	x_4	x_5
x_1		–	–	–	–
x_2	+		–	–	–
x_3	+	+		–	–
x_4	+	+	+		–
x_5	+	+	+	+	

Round 2. *Every participant broadcasts a value $g^{c_i y_i}$ and a knowledge proof for c_i, where c_i is either x_i or a random value $r_i \in_R \mathbb{Z}_q$, depending on whether participant P_i vetoes or not.*

$$g^{c_i y_i} = \begin{cases} g^{r_i y_i} & \textit{if } P_i \textit{ sends `1' (veto),} \\ g^{x_i y_i} & \textit{if } P_i \textit{ sends `0' (no veto).} \end{cases}$$

To check the final message, each participant computes $\prod_i g^{c_i y_i}$. If no one vetoes, we have $\prod_i g^{c_i y_i} = \prod_i g^{x_i y_i} = 1$. This is because $\sum_i x_i y_i = 0$ (Proposition 1). Hence, $\prod_i g^{x_i y_i} = g^{\sum_i x_i y_i} = 1$.

On the other hand, if one or more participants send the message '1', we have $\prod_i g^{c_i y_i} \neq 1$. Thus, the one-bit message has been sent anonymously.

Proposition 1. *For the x_i and y_i defined in AV-nets, $\sum_i x_i y_i = 0$.*

Proof. By definition $y_i = \sum_{j<i} x_j - \sum_{j>i} x_j$, hence

$$\begin{aligned} \sum_i x_i y_i &= \sum_i \sum_{j<i} x_i x_j - \sum_i \sum_{j>i} x_i x_j = \sum\sum_{j<i} x_i x_j - \sum\sum_{i<j} x_i x_j \\ &= \sum\sum_{j<i} x_i x_j - \sum\sum_{j<i} x_j x_i = 0. \end{aligned}$$

Table 1 illustrates this equality in a more intuitive way.

In the protocol, senders must demonstrate their knowledge of the discrete logarithms, namely the secrets x_i and c_i in each round respectively, without revealing them. This can be realized by using a zero-knowledge proof, a well-established primitive in cryptography [8]. Zero-knowledge proofs are commonly used in the related work in order to prevent certain attacks [11, 12, 13, 10]. Several zero-knowledge proof techniques have been presented in past literature [5,6,7,8]. One can use, for example, Schnorr's signature [7], which is suggested in Brandt's veto protocol [10]. Schnorr's signature is a suitable choice because it is short, non-interactive, and reveals nothing except the one bit information about the truth of the statement: "the sender knows the discrete logarithm" [7].

For example, let H be a *publicly known* secure hash function. To prove the knowledge of the exponent for g^{x_i}, one can send $\{g^v, r = v - x_i h\}$ where $v \in_R \mathbb{Z}_q$

and $h = H(g, g^v, g^{x_i}, i)$. This signature can be verified by anyone through checking whether g^v and $g^r g^{x_i h}$ are equal. One should note that here the participant index i is unique and known to all. Adding i inside the hash function can effectively prevent the replay of this signature by other participants. More details on Schnorr's signature and other zero-knowledge proof techniques can be found in [8,7].

There is a variant of our protocol, in which there is no need to use any zero-knowledge proofs. Instead, participants need to commit to their announcements before each broadcast round. This can be easily realized in the physical world — for example, all people write down their numbers on the paper before the broadcast round. However, in computer networks, this often requires additional rounds to send the results of applying a one-way hash function. It can prove costly if network communication is expensive.

2.2 Semantic Security

In order to analyze the security of our technique, we now examine the protocol more closely: in the first round, all participants announce their public keys g^{x_i}; in the second round, each uses a collaborative form of everyone else's public key to encrypt a one-bit message and announces the ciphertext.

To breach the anonymity of a participant, an observer — anyone within the broadcast range — may try to uncover the one-bit message from the announced ciphertext. In the following, we will prove that, under the DDH assumption, the proposed cryptosystem achieves *semantic security* [4]. This is equivalent to showing that under the hard-problem assumption, ciphertext is indistinguishable to observers [4]. First, we need to evaluate the resistance of our protocol against collusion.

Definition 2. *In a collusion attack, a subset of the participants are compromised, with their secrets x_i revealed.*

The *full collusion* against P_i involves all other participants in the network. Any anonymous veto protocol, by nature, cannot preserve the vetoer's anonymity under this circumstance. However, in practice, it is impossible to have all participants — who are mutually mistrustful — colluding against just one in an anonymous network; there would be no point for that person to stay in the network [1]. Hence, a more realistic attack is the *partial collusion*, which involves only some of the participants.

In AV-nets, the value of y_i is determined by the private keys of all participants except P_i. The following lemma shows its security property.

Lemma 3. *In AV-nets, y_i is a secret of random value to attackers in partial collusion against the participant P_i.*

Proof. Consider the worst case where only P_k $(k \neq i)$ is not involved in the collusion. Hence x_k is uniformly distributed over $\mathbb{Z}_q$ and unknown to colluders. The knowledge proofs required in the protocol show that all participants know their private keys. Since y_i is computed from x_j $(j \neq i, k)$ known to colluders plus (or minus) a random number x_k, y_i must be uniformly distributed over $\mathbb{Z}_q$. Colluders cannot learn y_i even in this worst case.

Theorem 4. *Under the Decision Diffie-Hellman assumption, attackers in partial collusion against P_i cannot distinguish the two ciphertexts $g^{x_i y_i}$ and $g^{r_i y_i}$.*

Proof. The secret x_i is chosen randomly by P_i. Lemma 3 shows that y_i is a random value, unknown to attackers. DDH states that one cannot distinguish between $g^{x_i y_i}$ and a random value in the group such as $g^{r_i y_i}$ [3].

The above theorem states that attackers cannot break the anonymity of the individual participant without full collusion. The one-bit message on the veto decision is decoded from the multiplication of all ciphertexts. The question is, whether additional information could be decoded as well. In our protocol, since the vetoer knows his random input, it is possible that he could derive the extra information: whether or not he is the only one who vetoed. If this is of much concern, there are solutions proposed in [13]. However, the derived information is only *one bit* and tells nothing about who else vetoed, nor how many vetoers there are. For this reason, this issue is generally not considered in the related work [1,13,12] — for example, a collision in the DC-net "leaks" the information that an even number of participants are sending messages, but that is not seen as a threat [1].

2.3 Attacks

Collusion is a common attack against anonymity [1,11,12,13,10]. Our protocol provides the strongest protection against such an attack — only full collusion can breach the anonymity of message senders.

Another attack makes use of message collisions. In the broadcast round of a DC-net, each participant sends one bit: b_i. The anonymous message received by everyone is the XOR of all the sent bits [1]. A known weakness in DC-nets is that an even number of messages would cancel each other out, forcing retransmissions [1]. Collisions not only reduce the transmission efficiency, but also can be exploited by attackers to jam the sent messages. For example, the last participant (an attacker) can announce $\sum_{i=1}^{n-1} b_i \mod 2$. Then, the overall message will always be '0'. A retransmission cannot resolve this problem as long as the attacker is the last announcer.

This collision attack may be viewed as one instance of disruption. Chaum suggested a few countermeasures to prevent disruptions [1]. Those relevant to this particular attack are twofold: broadcasting simultaneously on different frequencies or committing to output before broadcast [1]. However, both methods require additional rounds, which would significantly reduce the protocol efficiency.

In contrast, our protocol is resistant to collisions, whether intentional or not. First consider the situation where more than one participant sends the "veto" message. Each one randomly chooses r over $\mathbb{Z}_q$. Given a cyclic group with big q (*e.g.*, 1024-bit), the likelihood of message collisions, which results in $\prod_i g^{c_i y_i} = 1$,

Table 2. Comparison to the past work

related work	pub year	round no	broad-cast	pvt ch	colli-sion	3rd pty	collu-sion	security reliance	total traffic	total comp
GMW [11]	1987	$O(1)$	yes	yes	no	no	half	trapdoor	$O(n^2)$	$O(n^2)$
Chaum [1]	1988	2+	yes	yes	yes	no	full	uncond	$O(n^2)$	$O(n^2)$
KY [12]	2003	3	yes	no	no	yes	full	DDH	$O(n^2)$	$O(n^2)$
Groth [13]	2004	$n+1$	yes	no	no	yes	full	DDH	$O(n)$	$O(n)$
Brandt [10]	2005	4	yes	no	no	no	full	DDH	$O(n)$	$O(n)$
AV-net	—	**2**	**yes**	**no**	**no**	**no**	**full**	**DDH**	$\boldsymbol{O(n)}$	$\boldsymbol{O(n)}$

is negligible. In addition, intentional collisions are prevented by our protocol. Let $z = \prod_{i=1}^{n-1} g^{c_i y_i}$. The last announcer cannot send $1/z$ to jam the veto message — to provide the required knowledge proof for c_i, he would have to solve the Discrete Logarithm problem $(g^{y_i})^{c_i} = 1/z$, which is believed to be intractable [3].

3 Performance

There are related techniques proposed in past literature. Table 2 presents a comparison between our protocol and the previously proposed solutions.

Let us first compare AV-nets with DC-nets. Both protocols determine the boolean-OR of bits from a group of participants in such a way that message senders are untraceable. DC-nets could be implemented with different topological designs. A fully-connected DC-net is unconditionally secure. But it suffers from the scalability problem when applied to a large system. For this reason, Chaum suggests a ring-based DC-net in [1], which presents a trade-off between security and system complexity. Recently, Wright, Adler, Levine and Shield showed that the ring-based DC-net described by Chaum (also by Schneier [17]) is easily attacked [14]. They compared different topologies of DC-nets and concluded that the fully-connected DC-net is most resilient to attacks [14]. Hence we compare AV-net only with the most secure form of DC-net, *i.e.*, the fully-connected one.

A DC-net has two phases of operation: key setup and a one-round broadcast. The key setup phase — which produces $O(n^2)$ keys — is usually the problematic part in practice. In the original description of a DC-net, shared keys are established by *secretly* tossing coins behind menus. However it requires multiple rounds of interaction between pairs of participants. It is very slow and tedious, especially when there are many people involved. Other means to establish keys, as suggested by Chaum, include using optical disks or a pseudo-random sequence generator based on short keys [1]. However, such methods are acknowledged by Chaum as being either expensive or not very secure [1].

Our protocol replaces the problematic key-setup phase in a DC-net with a simple one-round broadcast. This is achieved via public key cryptography. Although a DC-net can adopt a similar technique — the Diffie-Hellman key exchange protocol — to distribute keys, its use of the underlying technology is quite different from ours. Suppose a DC-net uses Diffie-Hellman to establish keys[1]. Each participant must perform $O(n)$ exponentiations in order to compute the shared keys with the remaining $n-1$ participants. However, our protocol requires only one exponentiation for each of the two rounds. The computational load for each participant remains unchanged even when applied to a larger system (the cost of multiplication is negligible as compared to that of exponentiation).

Secure circuit evaluation is an important technique for secure Multi-Party Computation (MPC) applications. It evaluates a given function f on the private inputs $x_1, \ldots, x_n$ from n participants. In other words, it computes $y = f(x_1, \ldots, x_n)$, while maintaining the privacy of individual inputs. At first glance, it appears trivial to apply this technique to build a veto-protocol; one only needs to define f as the boolean-OR function. However, this general technique proves to be unnecessarily complex and expensive for solving a specific function like the Boolean-OR [10].

Yao [18] first proposed a general solution for the secure circuit evaluation for the two-party case. Later, Goldreich, Micali, and Wigderson extended Yao's protocol for the multiparty case, and demonstrated that any polynomial-time function can be evaluated securely in polynomial time provided the majority of the players are honest [11]. This conclusion is drawn based on the general assumption of the existence of a trap-door permutation function. Although the general solution proposed in [11] uses an unbounded number of rounds, it was later shown that such an evaluation can be done using only a constant number of rounds of interaction [15]. Recently, Gennaro, Ishai, Kushilevitz, and Rabin showed that three rounds are sufficient for arbitrary secure computation tasks [16].

Although the GMW solution to the circuit evaluation problem is more versatile than ours, it is much less efficient when used in a veto protocol. First, the GMW protocol requires pairwise private channels among participants [11], which has the complexity of $O(n^2)$. Second, it is no longer resistant to collusion when more than half of the participants are compromised. In such a case, the colluders can easily breach the privacy of other inputs. Third, it requires a large amount of traffic. Although the protocol could be completed with only three rounds [16], note that each round includes not only the broadcast of public messages, but also the transmission of private messages to everyone else through the pairwise secrecy channels [16]. The total amount of sent data is $O(n^2)$.

Kiayias and Yung investigated the Distributed Decision Making problem, and proposed a 3-round veto protocol [12]. They used a third party — a bulletin board server — to administer the process. The bulletin board server is a common way to realize a reliable broadcast channel. However, the server is needed for some other reasons. In the Kiayias-Yung protocol, each participant publishes $O(n)$ data. The final result on the veto decision is computed from $O(n^2)$ data. In large networks, it would be too demanding for individuals to store and compute such data. The server is a natural choice to perform the intermediary processing.

[1] Note that in this case, a DC-net is no longer unconditionally secure, as the Diffie-Hellman key exchange essentially rests on the Decision Diffie-Hellman assumption [3].

Groth modified the Kiayias-Yung veto protocol in order to reduce the system complexity [13]. His approach is to trade off round-efficiency for less traffic and computation. As a result, Groth's veto protocol allows each participant to publish a smaller amount of data, but requires participants to send their messages one after another, as one's computation depends on the result sent by the previous participant. Hence, instead of finishing the protocol in 3 rounds as in [12], Groth's veto protocol requires $n + 1$ rounds, where n is the number of participants.

Brandt studied the use of ElGamal encryption techniques for multiparty computation applications, and gave a 4-round veto protocol [10]. The performance of his solution, among others, is the closest match to ours. Its main disadvantage, however, is that it requires four rounds while ours only needs two. The difference in rounds lies in the way the veto messages are encrypted.

In Brandt's veto protocol, the first round is the same as in AV-nets: all participants broadcast their public keys. It requires one exponentiation to compute a public key. In the second round, each participant applies the standard ElGamal encryption algorithm to encrypt an explicit message: "veto" or "non-veto". Such an encryption requires two exponentiations. The third and fourth rounds are arranged to decrypt the messages, while preserving the privacy of individual inputs. It requires two and one exponentiations in each round respectively. Without taking the knowledge proofs into consideration, each participant needs to performs six exponentiations in total.

The novelty of our protocol is that the veto message is encrypted in a very implicit way (*i.e.*, by raising a base to one of two different powers). As a result, the veto decision can be immediately decoded after the second broadcast. It requires only two exponentiations in total, as compared to six in Brandt's protocol. Besides computational load, the traffic generated is also far less in our protocol, due to fewer rounds.

4 Conclusion

In this paper, we propose the Anonymous Veto Network (or AV-net) to solve the dining cryptographers problem. Several solutions in past work are reviewed, ranging from DC-nets and circuit evaluation techniques proposed nearly twenty years ago, to several anonymous ("private") veto protocols published in recent years. We show that our solution achieves semantic security and that the anonymity of message senders is preserved unless all other participants are compromised. In comparison with other methods, AV-net is more efficient in many aspects. It does not require any private channels or third parties; it has no message collisions, hence requires no retransmissions; it needs only two rounds of broadcast, fewer than any other solution; and the required computational load and bandwidth usage per participant are the least among the related work. Furthermore, there is very little room for improvement in each of these aspects. Its efficiency is close to the best we can possibly achieve under the security assumption of Decision Diffie-Hellman (DDH).

Acknowledgments

We thank Markus Kuhn for constructive discussions. We thank Ross Anderson, George Danezis, Mike Bond, Saar Drimer, Tyler Moore and other members in the Security Group for providing helpful comments and feedbacks. Special thanks go to Lihong Yang, Nick Wolfgang and Rachel Wolfgang for helping improve the readability of this paper.

References

1. Chaum, D.: The dining cryptographers problem: unconditional sender and recipient untraceability. Journal of Cryptology 1(1), 65–67 (1988)
2. Golle, P., Juels, A.: Dining Cryptographers Revisited. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 456–473. Springer, Heidelberg (2004)
3. Boneh, D.: The decision Diffie-Hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
4. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28, 270–299 (1984)
5. Chaum, D., Evertse, J.H., Graaf, J.V.D., Peralta, R.: Demonstrating possession of a discrete log without revealing it. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 200–212. Springer, Heidelberg (1987)
6. Chaum, D., Evertse, J.H., Graaf, J.V.D.: An improved protocol for demonstrating possession of a discrete logarithm and some generalizations. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 127–141. Springer, Heidelberg (1988)
7. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
8. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms. Technical report TR 260, Department of Computer Science, ETH Zürich (March 1997)
9. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
10. Brandt, F.: Efficient cryptographic protocol design based on distributed El Gamal encryption. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 32–47. Springer, Heidelberg (2006), http://www7.in.tum.de/~brandtf/studies.shtml
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proceedings of the nineteenth annual ACM Conference on Theory of Computing, pp. 218–229 (1987)
12. Kiayias, A., Yung, M.: Non-interactive zero-sharing with applications to private distributed decision making. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 303–320. Springer, Heidelberg (2003)
13. Groth, J.: Efficient maximal privacy in boardroom voting and anonymous broadcast. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 90–104. Springer, Heidelberg (2004)
14. Wright, M., Adler, M., Levine, B.N., Shields, C.: The predecessor attack: an analysis of a threat to anonymous communications systems. ACM Transactions on Information and Systems Security (TISSEC) 7(4) (2004)

15. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols. In: Proceedings of the twenty-second annual ACM Symposium on Theory of Computing, pp. 503–513 (1990)
16. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: On 2-round secure multiparty computation. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 178–193. Springer, Heidelberg (2002)
17. Schneier, B.: Applied Cryptography. J. Wiley and Sons, Chichester (1996)
18. Yao, A.: How to generate and exchange secrets. In: Proceedings of the twenty-seventh annual IEEE Symposium on Foundations of Computer Science, pp. 162–167 (1986)

A 2-Round Anonymous Veto Protocol
(Transcript of Discussion)

Feng Hao

University of Cambridge

The Chancellor is making a speech in the Galactic Security Council, listen up everyone, he says, I propose we should send troops to that enemy planet and occupy it; now is the time for the security council to take a vote.

You are the members of the security council, each one of you has the veto power, and it's time for you to cast your vote. The setting of this problem is that you don't have any private channels, the only way for you to express your opinion is through a public announcement, however, if you publicly declare that you want to veto you would have offended some people, and you will be punished. So the question is, if everything is public, and everything you say, or you send, can be traced back to you as the data origin, how do you send an anonymous message in such an environment. It is mind-boggling this is possible in the first place, but with public key cryptography it is possible. In my talk I am going to present a solution to this puzzle, in addition I will show that the solution is very efficient in almost every aspect.

Frank Stajano: I don't get why you are listing terms with different indices if the summation is over x_i, y_i where the indices are the same?

Reply: No, the y_i expression is in terms of all the x values, *except* x_i. This is the secret aspect of this protocol. The protocol achieves semantic security because, based on the Decision Diffie-Hellman assumption, it is compromised only under full collusion case. In other words, if the Chancellor wants to find out who has vetoed, basically he has to compromise everyone in this room to get the individual private key, which is quite unlikely. And the protocol is resistant to disruptions, so the Chancellor can be involved in the voting as well, but he cannot suppress other people's veto. If he wants to do that, he has to solve the discrete logarithm problem, which is believed to be intractable.

If we compare with the other related techniques, the one technique which is the closest match in terms of performance to ours, is Brandt's protocol. However, the biggest disadvantage in his protocol is that he needs four rounds, but for our protocol we only need two rounds. Why the difference? The difference is because in Brandt's protocol he used the standard El Gamal encryption, the first round is exactly the same as our protocol, and the second round, each participant uses the El Gamal encryption to encrypt the veto message, and it requires two additional rounds to decrypt it securely. But for our protocol, we use some unconventional ways to encrypt the message. We encrypt the message by raising our value to the power of two different values, and the advantage of this approach is that the veto message can be decoded immediately after the second round. Also Brandt's protocol has the same system capacity as ours, but because of the difference of constant factors hidden in their O-notation, the actual amount of computation nodes and the bandwidth in his protocol are actually several times more than those in our protocol.

Finally I bring the conclusion. We propose an anonymous veto network, or AV-net, to solve the veto problem, or the dining cryptographers problem. We don't need any secret channels, no third parties, and have no message collisions, it is provably secure on the Decision Diffie-Hellman assumption, and the efficiency is close to the best we can possibly achieve. We think it is quite unlikely there could be any other solutions more efficient that ours. That's all for my talk. Any questions?

James Heather: Have you proof for the claim of maximum efficiency?

Reply: If we look at the protocol, in the first round you only need one exponentiation for each party, that's all, and in the second round, you also need to do just one exponentiation. What less can you get? And for the bandwidth usage, in the first round, the data size you need to send is the underlying group size, and for the second round is also the minimum data size we can possibly achieve. It may be you can get slightly more efficient, but it's quite unlikely. Yes, if you think that is a proof.

Bruce Christianson: If you want to veto, is it important that you raise the x and y_i to some random power, or can I just send a random number?

Reply: No, because you need an honest proof for this. If you send the random numbers, and you are able to suppress the veto, that is why we need an honest proof here.

Bruce Christianson: And so it doesn't matter what order people go in the second round?

Reply: No.

Kenny Paterson: There's another interesting cryptographic primitive called ring signatures[1], have you ever come across this? I guess the issue is that they don't work if you can identify the actual initial sender of the message, you need some kind of mixnet ring signatures that I can place in a computer as coming from me, or from you. If you could put in a ring signature which contains a veto message that could have been signed by any one of a group of people, then it has similar functionalities.

Reply: Yes, I think that is a different underlying assumption, because for this protocol we have very clear authenticated broadcast channel. That means, if you send the message then the people will know exactly it's you who sent the message.

[1] How to leak a secret, Ron Rivest, Adi Shamir, and Yael Tauman, ASIACRYPT 2001, LNCS 2248.

Kenny Paterson: So it's the same solution but with different network assumptions?

Reply: Yes, for the different scenarios, and different cases.

Bruce Christianson: Suppose we're voting on whether to admit someone to our club, and it requires two no votes to blackball, can we generalise this approach in that way?

Reply: In that case I think the general way of doing that is using the secret sharing. You can have, for example, three out of five secret sharing if two people decide not to give any secret sharing, effectively they are doing the vetoing.

Bruce Christianson: But then it's reasonably clear who hasn't given their shares? Which is what you want to avoid.

Reply: Yes, that's true, yes, that's a quite interesting question, I'll look into that.

Craig Gentry: I think you could just do, essentially two of your protocols, and then your proof would be basically that you'd provided proof of knowledge that proves that you didn't do two random values. So you can produce at most one random value, and the other value has to be equal to your x_i, can you do a proof like that efficiently, do you see what I'm saying? In order to accomplish what Bruce wants, you would just basically do a proof of knowledge, in which you would have two values in the group, and you would prove that you have randomised, at most, one of the values, so in other words, umm, actually this doesn't quite work.

Bruce Christianson: Yes, it's a good step though.

Craig Gentry: It's the beginning of a solution, but the problem would be what if the two different objecting people randomised the same thing, one veto would be masked.

Richard Clayton: You can do a slot reservation rather like the dining cryptographers protocol.

Reply: I think that's possible, but on the other hand it is also quite easy to detect. In the full paper we say it's important for the public keys to be all different, you cannot have duplicate. In theory, yes, but in practice, it's just too easy to detect.

Bruce Christianson: Well there's a very small chance that you'll get nobody vetoing by chance in fact.

Reply: You mean the collision? It's extremely unlikely.

Bruce Christianson: Well perhaps we can try and work out how to do a two veto vote.

Reply: Yes, I think that one is future research.

How to Speak an Authentication Secret Securely from an Eavesdropper

Lawrence O'Gorman, Lynne Brotman, and Michael Sammon

Avaya Labs, Basking Ridge NJ 07920, USA
{logorman,lynne,mjps}@avaya.com

Abstract. When authenticating over the telephone or mobile headphone, the user cannot always assure that no eavesdropper hears the password or authentication secret. We describe an eavesdropper-resistant, challenge-response authentication scheme for spoken authentication where an attacker can hear the user's voiced responses. This scheme entails the user to memorize a small number of plaintext-ciphertext pairs. At authentication, these are challenged in random order and interspersed with camouflage elements. It is shown that the response can be made to appear random so that no information on the memorized secret can be learned by eavesdroppers. We describe the method along with parameter value tradeoffs of security strength, authentication time, and memory effort. This scheme was designed for user authentication of wireless headsets used for hands-free communication by healthcare staff at a hospital.

1 Introduction

When we type a password into a computer or a PIN into a bank machine, the characters are usually masked on the screen (*e.g.*, "******") to prevent onlookers from seeing the secret code. When speaking a password or PIN into a telephone or other voice communication device, there is nothing comparable to keep the password secret. There are two alternatives for the user. One is to speak softly and to hope no eavesdropper hears. Another is to use a telephone keypad and to make sure that no onlooker sees which keys are pressed. Neither solution is very satisfactory. The former depends on the user finding a place out of earshot from others. The latter precludes hands-free communications and fails to offer a solution for present and future communications devices that use only voice communications. The objective of this work is to offer a means for user-authentication by voice in which an eavesdropper who can hear the user responses cannot gain information to impersonate the true user.

We propose a method called SPIN, for "spoken PIN". This authentication protocol involves the user first memorizing simple plaintext-ciphertext pairs, *e.g.*, red=3, green=2, blue=9. A plaintext sequence is sent – spoken – by the authentication server to the user. We assume the user receives this sequence by an earphone or in some way that eavesdroppers cannot hear it. The user responds with the ciphertext element associated with each plaintext element. We assume there may be eavesdroppers hearing this response. We show that by randomizing the sequence of authentication elements and by interspersing *camouflage* elements in the sequence, that an eavesdropper can obtain no information to learn the cipher over one or many responses.

In Section 2, we describe some alternative authentication methods where the user response can be seen or heard by eavesdroppers and the procedure can still be secure. In Section 3, we detail the SPIN method and provide equations and plots to choose among parameter value tradeoffs. We also make some security and convenience comparisons between SPIN and PINs, passwords, and one-time passwords. We describe our healthcare application of SPIN in Section 4 and discuss SPIN and some alternatives in Section 5.

2 Background

There is a common solution to any authentication problem where eavesdroppers may hear or see the user's response. This is not to repeat that response. Or, more specifically, the correct authentication response will change randomly each time it is given. This is different than a traditional, static password. One reason the changing authentication response protocol is less common than the static password protocol is that, for a changing response the user usually requires some sort of aid, such as an electronic token, to correctly respond with a different response for each authentication instance. One-time passwords, time-synchronous PINs, and challenge-response pass codes are all examples of this.

When one speaks of spoken authentication, one might picture a secret rendezvous between spies where they identify themselves by a pre-arranged dialogue such as,

Spy 1 – *"I hear Cyprus is lovely this time of year."*
Spy 2 – *"The leeward beaches are best in the afternoon sun."*

As much as a dialogue such as this might help the two spies identify each other, it can only be used once because an eavesdropper who has heard the dialogue can impersonate either spy subsequently. This is akin to a one-time password scheme of which there are computer authentication examples such as SKEY, which is a chained list of hashes that the user carries and uses one each time until the list is depleted [1, 2]. Because each password is used once, it would be tedious for most users to memorize the list, so instead it is carried with the user, either in paper form or on a portable electronic device.

Time-synchronous pass codes work in the following way [3]. A user has an electronic token that generates a different number periodically, say every minute. The server to which the user is authenticating generates the same number sequence synchronously to the token. Therefore, when the user wants to authenticate, she sends the current pass code to the server for confirmation that it is the current number. Even if an eavesdropper sees the number, this will not be useful in subsequent authentication attempts because the sequence changes randomly.

A challenge-response pass code is similar to the time-synchronous one in that a user must also carry an electronic device. In this case, the user first requests to authenticate to the server. The server sends a random number challenge. The user enters the random number into the device, which performs cryptographic calculations to generate a response pass code that is sent back to the server.

The Query-Directed Password (QDP) scheme [4, 5] is a challenge-response scheme using multiple-choice, personal questions, such as, "What was the color of the car on which you learned to drive? 1) black, 2) white, 3) blue, 4) red, 5) green, 6) gray." If the user responds with the number of the multiple-choice answer for each question, and if the questions and/or the numbers associated with answers are randomized between authentication instances, then an eavesdropper will hear a different sequence of numerical responses each instance. Implemented correctly, responses will appear to an eavesdropper as random numbers. QDP uses a number of personal questions, or challenge questions, that are more varied and secure than the common "mother's maiden name" challenge [6-8]. The user does not have to memorize anything because answers to the personal questions will already be known (if chosen well). However, there are some drawbacks to this approach. One is a time-security tradeoff. It takes time to read each question including multiple choices. Furthermore, more questions and/or more answer choices are required for higher levels of security (4-5 questions of 6 multiple choices each were used in our testing). We compare this QDP approach with the proposed method of this paper in Section 3.5.

Our voiced-password problem would seem perfectly suited for a speaker verification solution. However, for our application, which involved health care workers speaking over a headset in an often noisy hospital environment, the recognition rate (or verification rate) is not reliable enough at this time.

There is an analogous user authentication procedure that also involves a one-sided eavesdropper. This is a graphical password scheme that can defend against a Trojan keyboard logger (*e.g.*, [9]). A keyboard logger program that has been secretly installed on a victim's computer can record all keystrokes for an attacker to learn passwords. A graphical password scheme can defend against this in the following way. Pictures are displayed on the screen with identifying numbers, but these numbers change each authentication session. The user is asked to enter the numbers associated with the pictures she has memorized for authentication. Analogously to an eavesdropper hearing a spoken authentication response for our application, this logger can capture the authentication numbers that are entered, but because the numbers change each time, no useful authentication information is gained.

Passwords, challenge questions, and biometrics involve human factors such as memory and physiology, as well as security considerations. This combination – often a tradeoff – between human factors and security is also important to security protocols where the human is involved. Security protocols involving humans have been used since the Egyptian, Greek, and Roman empires (and before) [10]. New human security protocols are being proposed and analyzed with current-day knowledge of computer security [11]. SPIN is a user-authentication scheme that falls into this category of human security protocols because a human performs her half of the protocol without aid of a computing device.

3 SPIN–Spoken PIN

The SPIN method is described first by example in Section 3.2. Although the method itself is straightforward, it is not so obvious how to optimize the security-time-memorization tradeoffs. Section 3.3 describes parameters of the method and we examine the tradeoffs in Section 3.4 with respect to these parameters. We begin with some definitions.

3.1 Definitions

The solution presented here to securely speak an authentication secret involves the use of a substitution cipher, a challenge-response protocol, and camouflage elements. We define these terms here.

A *substitution cipher* is a secret code where a sender substitutes characters of plaintext in a message by characters of ciphertext, and sends this coded message to the receiver. The receiver who knows the cipher inverts the substitutions to recover the plaintext. A simple substitution cipher might substitute each character by the character one above in the alphabet (the simplest form of Caesar cipher [10]). The secret describes the substitution rule(s) and only the sender and authorized receiver(s) should share this knowledge.

A *challenge-response protocol* used in user-authentication is an exchange between an authentication server and the authenticating user whereby the server sends a message to the user that changes each time, and the user returns a response that depends upon the challenge. A primary advantage of challenge-response protocols for user-authentication is that an attacker cannot simply replay the response, because it is different for each challenge. In this paper, we refer to the exchange between server and user as a *challenge-response sequence*, because there is not just one challenge and one response, but a sequence of these that make up a single authentication session.

An *element* is one component of a sequence. This element can be a character, word, or number. In this paper, challenge elements are colors, and response elements are numbers, but either could equally well be letters, words, animal names, names of planets, etc. An *authentication pair* consists of a single challenge element and its corresponding response element. An *authentication element* is a general term for either a challenge element or a response element, in other words it is just one of the elements that is used for authentication. In contrast, a *camouflage element* is one that is not used for authentication. It is interspersed with authentication elements to reduce the chance that an eavesdropper will learn the substitution rules after hearing one or more authentication sequences. In the next section we use the convention of bold font for authentication elements to distinguish these from camouflage elements in regular font.

We describe two major attackers. One is the *eavesdropper* who we will call *Eve*. She can hear user responses. We assume we have no control on Eve, so she can hear an unlimited number of responses to try to gain authentication. The other attacker is *Brutus*, who can steal the headset in some brute force manner, then *hear authentication challenges* and try an exhaustive guessing approach. In addition, Brutus can learn from listening to repeated challenges, if given the chance to do so, to mount a more sophisticated attack. We have some control on Brutus because we know when he answers erroneously. Since each element is challenged and responded to individually, we have the ability to respond to erroneous elements by, for instance, freezing the account. So, Brutus's freedom to attack is much more limited than Eve's. These two attackers can collude.

The description of the two attackers above also defines the *threat model* that we examine in this paper. The SPIN threat model is alike that of passwords and PINs in most ways. Like these traditional authentication methods, a user's SPIN code must be memorized and kept secret from attackers. If lost, an attacker can gain access to the user's account. If forgotten, the user cannot authenticate. SPIN is different than passwords and PINs in one major way: there is no threat if the SPIN response is learned by attackers. Just like passwords and PINs, a more comprehensive SPIN threat model also includes issues such as security of the channel (in this case the wireless channel), protection of the secret at the server, etc. But, it is the threat from Eve and Brutus that we examine in depth here.

Finally, we define the term *security strength* as being the total number of attempts an attacker must make to be sure to find the authentication response: the more attempts required, the greater the security strength. For this paper, the security strength is generally the number of permutations an authentication challenge can take. (Note that a brute force guesser will likely guess the answer in half the number of permutations on average.)

3.2 Description by Design Progression and Example

The SPIN method is a straightforward substitution cipher into which we intersperse camouflage elements. We describe it in this section by a progression of methods, starting with the most straightforward and adding modifications to address shortcomings, finally leading to the proposed method. In the following section, we generalize the method.

Method a – First, consider a simple substitution cipher where colors are replaced by numbers. For instance,

Substitution rules: **3** → **Red**, **2** → **Green**, **9** → **Blue**, **6** → **Yellow**.

An authentication session using the rules above might look like,

Challenge from server: **Blue, Red, Yellow, Green**
Response from user: **9, 3, 6, 2**
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

Since the challenge changes each time, Eve cannot merely hear **"9, 3, 6, 2"** and replay it to successfully authenticate. So, we have made one step toward the goal of speaking a password without an eavesdropper being able to decipher and repeat it.

However, there is a problem with this simple cipher method. If Eve hears a few responses, even though the elements will be in different order, she will soon learn that the cipher code only uses elements whose numbers are **{2, 3, 6, 9}**. Therefore the number of different permutations that these digits might take, is reduced from $10 \times 9 \times 8 \times 7 = 5040$ for any 4 different digits randomly ordered, to $4 \times 3 \times 2 \times 1 = 24$ for these 4 specific digits.

Method b – To strengthen the security, the server can randomly intersperse camouflage elements in the challenge sequence. Using the same substitution rules as above, an example of an authentication session is,

Challenge from server: 1, 8, **Blue**, 4, **Red**, **Yellow**, 0, 7, **Green**, 5
Response from user: 1, 8, **9**, 4, **3**, **6**, 0, 7, **2**, 5
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

The user performs the substitutions only for the colors and simply repeats the camouflage elements. The server needs only to check that the substitutions are correct and that the camouflage elements have been repeated. The camouflage is present only to prevent Eve from learning the authentication elements in the response. Since, as can be seen in the example, there are 10 different digits in the response, Eve will always hear some different ordering of 10 digits and will not learn anything about the cipher.

Although Eve does not gain information from these permutations of digits 0-9, Brutus can gain information from hearing the challenge sequence. After hearing an entire sequence, Brutus knows that the color substitution values are all the numbers that were not present in the challenge. So, this again reduces the permutations from $10 \times 9 \times 8 \times 7 = 5040$ to $4 \times 3 \times 2 \times 1 = 24$.

Method c – Instead of inserting camouflage elements that are dependent upon the authentication elements, such that all digits from 0-9 are present exactly once in a sequence, let's try choosing camouflage elements randomly. The larger the number of camouflage elements we choose, the greater is the chance that one or more will overlap with the values of the authentication elements. When this happens, Brutus, who hears a challenge, will not be able to know all the color substitutions just by hearing the numbers not present in the challenge. In the following example, we've added 6 random camouflage elements,

Challenge from server: 1, 8, **Blue**, 9, **Red**, **Yellow**, 1, 7, **Green**, 3
Response from user: 1, 8, **9**, 9, **3**, **6**, 1, 7, **2**, 3
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

The color substitution values for blue and red happen to appear in the camouflage elements. Therefore, Brutus who depends on learning authentication elements by numbers that are not present in the sequence, will not learn that 9 and 3 are color substitution values. If we increase the number of this type of camouflage elements (to infinity), we increase the assurance that all color substitution values will be present so the attacker's information decreases (to zero).

However, by adding camouflage elements independent of authentication elements, we can succumb to an attack at the eavesdropper's end. If Eve listens to a few response sequences, she can learn that authentication elements, which are always there, occur with higher frequency than camouflage elements, which don't have to be there. We call this a histogram attack. Once Eve has heard enough responses to learn the authentication elements, this again degrades to an attacker needing only $4 \times 3 \times 2 \times 1 = 24$ guesses.

Method d – Since Methods b and c defend against Eve or Brutus separately, perhaps we can combine these approaches to yield method resistant to each. Start with authentication elements randomly ordered. Randomly insert camouflage elements chosen dependently as described in Method b. Then, randomly insert more camouflage elements independently as described in Method c. In the following, we've added 3 independently chosen camouflage elements to the 6 dependently chosen camouflage elements already there and the 4 authentication elements,

Challenge from server: 1, 8, **Blue**, 4, 9, **Red**, **Yellow**, 0, 1, 7, **Green**, 3, 5
Response from user: 1, 8, **9**, 4, 9, **3**, **6**, 0, 1, 7, **2**, 3, 5
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

Eve obtains no information from hearing the response sequence because she still hears a permutation of 0-9, but now there are added digits chosen independently of authentication elements, so this gives no additional information.

Now, Brutus who listens to the challenge will hear every digit except 2 and 6. So he knows that two color substitution values must be these digits, but has no information on the other two. If we increased the number of independently chosen camouflage elements, we would eventually include all color values (with some probability) such as to defend against Brutus (with some probability).

Method e – Instead of trying to hide the authentication elements in a probabilistic fashion as in Method d, we can do this deterministically by slightly modifying the way the challenge sequence is given. Instead of a challenge element being "number" or "color", each is, "number or color". When the color is not an authentication element, the user just ignores it and repeats the number. When the color is an authentication element, the user ignores the number and responds instead with the authentication number corresponding to the color. Following is an example of this combination,

Challenge from server: 1 or Purple, 8 or Black, 3 or **Blue**, 4 or Pink,
2 or **Red**, 6 or **Yellow**, 0 or Orange,
7 or Gray, 9 or **Green**, 5 or White
Response from user: 1, 8, **9**, 4, **3**, **6**, 0, 7, **2**, 5
Decipher by server: **9=Blue, 3=Red, 6=Yellow, 2=Green**

From this example, there are two types of challenge elements. One type is the camouflage element (*e.g.*, "1 or Purple") that contains a number and color, and neither the number nor color can be an authentication element. There is no correspondence between these camouflage colors and numbers, their pairings are random. The second type of challenge element is the authentication element (*e.g.*, "3 or **Blue**"). This contains a number and an authentication color. The number must be one of the authentication number responses, but there is no correspondence between an authentication number and color in a single element; indeed as can be seen in the example, "6 or **Yellow**", the number and color can be the correct authentication pair. No color or number can repeat in a challenge sequence.

Now, Brutus who listens to the challenge will hear all the possible digits and all the possible colors. So, he will not be able to ascertain which digits are authentication elements by their absence in the challenge sequence (as in Method c). Furthermore, Eve will always hear a randomly ordered sequence of every possible response repeated once, and once only. Therefore, she will not gain any information over an unlimited number of responses.

So, at the expense of a little more time to speak a more wordy challenge sequence, we have defended equally against both Brutus and Eve for a single authentication challenge-response. There is one small addition we make to this protocol. We said that the numbers spoken for authentication elements in the challenge sequence are random orderings of authentication numbers. This is true for any ordering except for the one where authentication numbers and colors all happen to correspond exactly, for which we make an exception. For the example above, this is a challenge sequence containing any ordering of all the pairings, "**9** or **Blue**", "**3** or **Red**", "**6** or **Yellow**", "**2** or **Green**". Although this random pairing has the same probability as any other, we do not use it as a precaution against the "naïve" attacker who just repeats all the challenge numbers and would then succeed to authenticate for this single case.

From this section, we have found two methods that offer security against both Eve and Brutus. Method d offers a probabilistic measure of security at the expense of additional camouflage elements. Method 3 offers a deterministic measure of security at the expense of more complexity in the protocol. In the next section, we generalize the approach and in Section 3.4 examine tradeoffs in security, time to authenticate, and memorization effort.

3.3 Formal Description

In general, the SPIN code can be described by the following parameters,

$$\begin{aligned}
\textbf{SPIN} \quad & (m, a, c_D, c_I, L) \qquad (1)\\
m &= \text{number of memorized substitution pairs}\\
a &= \text{number of authentication elements in a code}\\
c_D &= \text{number of dependent camouflage elements in a code}\\
c_I &= \text{number of independent camouflage elements in a code}\\
L &= \text{number of levels, or values, that elements can take.}
\end{aligned}$$

From Method b, the minimum number of elements in an authentication sequence is $n_{min} = a+c_D$. Because the c_D elements are chosen to be all the element values that are not authentication element values, then $c_D = L-a$, so $n_{min} = L$. From

Method d, we insert independent camouflage elements to the sequence, so the total number of elements in a sequence is,

$$\text{Total number of elements: } n = a + c_D + c_I = L + c_I. \tag{2}$$

From Section 3.2, the best- and worst-case security strengths are,

$$\text{Best case security: } S = L(L-1)(L-2)\ldots(L-a+1) = \tbinom{L}{a}\,a! \tag{3}$$

$$\text{Worst case security: } S = a!. \tag{4}$$

For Method e, there are no independent camouflage elements. So, $c_I = 0$ in equation (2) and the total number of elements in a sequence is, $n = a + c_D = L$. Because this method defends equally against Eve and Brutus, the security strength is the same for both and is the best-case security strength, equation (3), minus 1 due to the exception of excluding the case where challenge numbers and colors correspond, as mentioned above.

Method d is less straightforward than Method e because of the independent camouflage elements and the fact that security strength is affected by these elements probabilistically. The best-case security strength for Method d occurs for the case when all authentication elements are located at the beginning of the authentication challenge. In this case, Brutus obtains no information from the camouflage elements, because he has to make his guesses before learning the information he gains from hearing them. An example of a best-case challenge is, "**Blue, Red, Yellow, Green**, 0, 4, 7, 1, 8, 5".

The worst-case security strength occurs for the case when all authentication elements are located at the end of the authentication challenge. In this case, Brutus has obtained as much information as there is from the camouflage elements before having to guess the authentication elements. An example of a worst-case challenge is, "0, 4, 7, 1, 8, 5, **Blue, Red, Yellow, Green**".

When we add c_I elements, we can raise the worst-case security strength closer toward best-case. This is because some c_I elements might have the same values as authentication elements, thus preventing Brutus from learning about authentication element values through their absence. However, because the c_I elements are chosen independently of authentication element values, we can only say probabilistically whether there will be repeating elements. Obviously, the more the c_I elements there are, the greater the chance of repeating authentication elements. To understand the effect of adding c_I elements, we need to determine the number of elements, c_I, necessary to obtain a probability P that k or more of the a authentication elements ($k \leq a$) is repeated in the c_I elements.

The probability that $k = a$ authentication elements are present in c_I randomly chosen elements can be derived as follows:

$$P(k = a = 1, c_I) = 1 - ((L-1)/L)^{c_I}$$
$$P(k = a = 2, c_I) = 1 - 2\,((L-1)/L)^{c_I} + ((L-2)/L)^{c_I}$$
$$\ldots$$

This can be described by the forward difference operator, $D(f(x)) = f(x+1) - f(x)$,

$$P(k, c_I) = D^{c_I}\left((L-k)/L\right)^{c_I}$$

In general, the probability is,

$$P(k, c_I) = \sum_{i=0}^{k} (-1)^{k-i} \binom{k}{i} \left((L-k+i)/L\right)^{c_I} . \tag{5}$$

Besides the probability that *all* authentication elements are repeated in the independent camouflage elements, we will also ask the probability that *all or one less than all* authentication elements are repeated in the independent camouflage elements. This is,

$$P(k = a \text{ or } k = a-1, c_I) = P(k = a, c_I) + a\left(P(k = a-1, c_I) - P(k = a, c_I)\right) . \tag{6}$$

3.4 Security-Time-Memorization Tradeoffs

In practice, we want to optimize with respect to three parameter values. We desire high security strength, short authentication sequence (or session) length, and a small number of authentication pairs that a user has to memorize.

First, we simplify our choices by setting $m = a$. This implies that all the user's memorized elements are used in each authentication session. Although this does not have to be the case and there are security advantages of memorizing more pairs than are used each session, we do not explore this here, choosing instead to minimize the memorization effort of the user.

Secondly, for Method d, we make a choice for the probability value that affects the number of independent camouflage elements in Method d. We choose this to be, $P(k, c_I) = 66.6\%$. This choice is somewhat arbitrary, but we feel it to be practical at least for our own application described in Section 4. In words, this means that we are calculating the number of independent camouflage elements in which there is a two-thirds probability that a certain portion of authentication elements are repeated in these. We make another choice that this "certain portion" we calculate for will be all authentication elements, a, or all or one less than all authentication elements, a or $a-1$.

With equations (2-6), we can plot the security-time-memorization tradeoffs for Methods d and e. Figure 1 is for Method d for 66% probability that all or one less than all authentication elements will be in the independent camouflage elements. Figure 2 is for Method e.

We can obtain an idea of how these methods compare by examining a couple examples. For a requirement of $S = 1000$ and $m = 5$, we need about $n = 20$ for Method d in Figure 1, but only about $n = 6$ for Method e in Figure 2. For a requirement of $S = 100$ and $m = 4$, we need about $n = 15$ for Method d in Figure 1, but only about $n = 5$ for Method e in Figure 2.

Although we don't plot Method d for the 66% probability case that all authentication elements are in the c_I camouflage elements, we can obtain a comparison

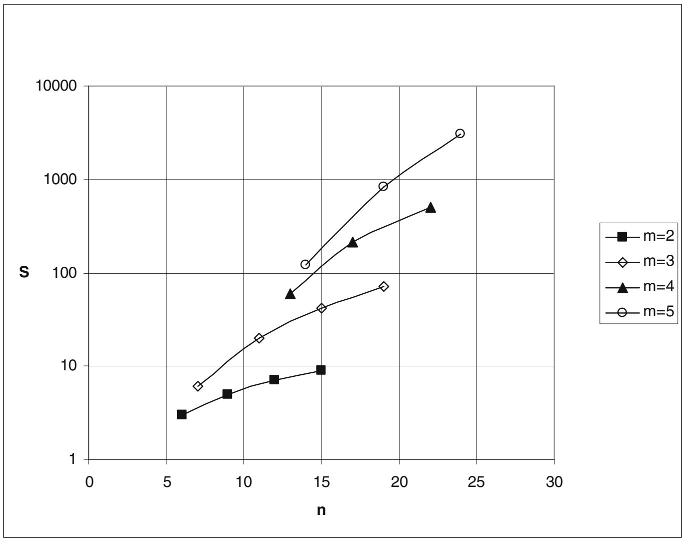


Fig. 1. Plot for Method d for 66% probability that all or all minus one authentication elements will be in the cI camouflage elements. Plot shows tradeoff among number of authentication elements the user must memorize, m, the security strength, S, and the length of the authentication sequence, n. The points are calculated for $L = \{4, 6, 8, 10\}$ on $m = 2$ and $m = 3$, $L = \{5, 6, 8, 10\}$ on $m = 4$, and $L = \{6, 8, 10\}$ on $m = 5$ (all from low to high n).

from the equations. For a requirement of $m = 4$ and $S = 1000$, Method d can attain this with $n = 24$. In Figure 2, Method e requires a far shorter sequence, $n = 7$. For a requirement of $S = 10,000$ and $m = 5$, we need about $n = 30$ for Method d, but only about $n = 9$ for Method e.

In these examples, the required sequence length was greater than 3 times for Method d than Method e. Although Method e is has a wordier challenge sequence, it takes only about 50% more time per challenge-response element for Method e than for Method d. Therefore, based on length of sequence (or time to authenticate) alone, Method d takes about twice as long, therefore is less desirable from this respect than Method e.

3.5 Comparing SPIN to PINs, Passwords, One-Time Passwords, and Challenge Questions

From Section 3.1, the security strength of a randomly-generated authentication response is the number of different possibilities it may take. For a PIN with 4 digits, the security strength is 10^4. For a Method e SPIN code we can achieve $S = 10^4$ with 5 authentication elements and a sequence length of 8, or with 4 authentication elements, we can achieve less security of 5040 with a sequence length of 10. In this example and in general, SPIN requires more memorization

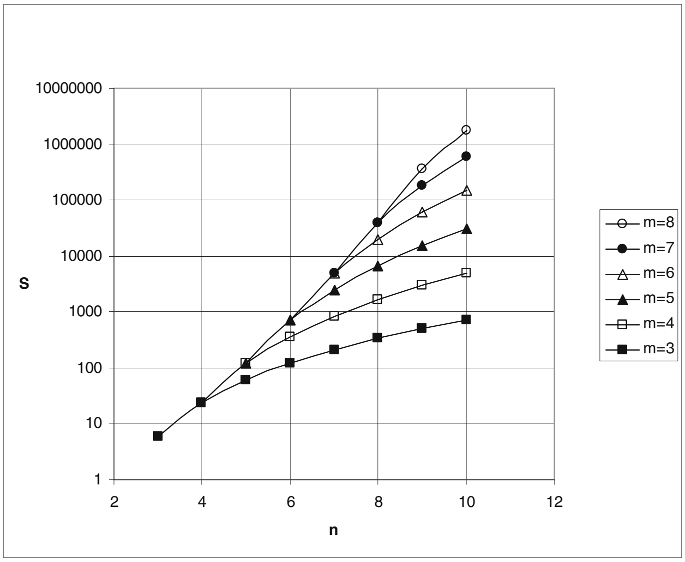


Fig. 2. Plot for Method e. Plot shows tradeoff among number of authentication elements the user must memorize, m, the security strength, S, and the length of the authentication sequence, $n = L$.

effort or a longer time to authenticate, or both, to achieve a similar level of security.

A password achieves much stronger security than a comparable-length PIN because there are many more choices of characters. For an 8-character password made up of any combination of upper and lower-case characters and digits, security strength is 62^8. One might initially suspect that we can gain such efficiency with SPIN as well by having many more levels (*i.e.*, more colors in our examples above) such as to increase the security strength. It is true this will increase the security strength, but there is a huge cost, as seen in equation (2), $n = L + c_I$. The total authentication sequence length must be equal to or larger than the number of levels of authentication elements. Even if only lower-case characters were considered, a 26-element authentication sequence would require too lengthy a response from the user.

As mentioned in Section 2, a one-time password from a list or token that the user carries will meet the needs of resisting an eavesdropper. Since a one-time password can contain any combination of characters and digits, it can have strong security per number of characters spoken. Therefore, this is a good alternative to SPIN — except for cases where we don't want to burden the user to carry a token or where we desire hands-free authentication.

QDP or challenge questions [4, 5], where the user responds with yes/no or multiple choice answers, can meet our specification for secure voiced authentication. The advantage is that these questions can be designed to be more easily remembered by the user. The disadvantage is the time required to achieve a reasonable level of security. For QDP, if 5 questions are asked with 6 multiple choices each, this takes about $5 \times 15 = 75$ seconds and yields a security strength of $6^5 = 7776$. A comparable Method e SPIN code is for $m = 5$, $n = 8$, $S = 6720$. At 3 seconds per challenge-response element, this takes about 24 seconds, less than one third the time of QDP.

It is clear that SPIN fares less well against password, PIN, and one-time password for each comparison in this section of security strength, sequence length and memory effort. However, because none of the traditional schemes meets the requirement of hands-free, spoken authentication, these disadvantages are the price to achieve this. The main tradeoff between SPIN and QDP is memory effort versus time to authenticate.

4 Use of SPIN in a Health–Care Application

The SPIN code was initially developed to meet a need of MACCS (Mobile Access to Converged Communications Service), a system initially targeted for use by healthcare workers. This is a wireless, voice-interactive, hands-free communications system. Each user has a wireless headset for issuing voice commands to the system and communicating with other users. To protect users' and patients' private information that is communicated on MACCS, the users must authenticate themselves to the system. A static password or PIN is not acceptable because it can be overheard by eavesdroppers, and a one-time password requiring a token or list is unacceptable because it is not hands-free. QDP (as described in Section 2) was the initial authentication scheme used in MACCS, however the 3-4 questions took over 1 minute per authentication session. To reduce authentication time, SPIN was provided as an alternative to QDP.

Initial response to the use of SPIN was mixed. As compared to the QDP method, users were happy to authenticate more quickly. However, they were not happy to memorize another security code. We will report on our experiments and their results in a later paper.

5 Summary and Discussion

In this paper, we have described two variants of a method for speaking a password securely, potentially in front of eavesdroppers. The method involves the user memorizing color-number pairs. A challenge is sent to the user involving camouflage numbers and authentication colors. The user repeats the camouflage numbers and substitutes colors for the corresponding, memorized numbers. We have described a protocol whereby this authentication scheme can be secure from eavesdroppers and brute force attackers. There seems to be little or no treatment in past literature on this topic. However, while working and testing QDP and SPIN, other spoken authentication methods have arisen. We have not performed user testing with these, so we merely mention these alternatives as potential for future work.

One method decreases the time (number of elements) by requiring the user to perform some elementary mathematics. The user memorizes m digits. A challenge consists of m randomly chosen digits. The user responds with the addition of each challenge digit to a memorized digit, modulo-10. This eliminates the need for camouflage elements because the response that Eve hears will be random.

Another method reduces the memory effort. Instead of using random color-number pairs, more memorable pairing schemes can be used. Colors can be listed by the user's order of preference, or country names can be used by the user's order of preference for vacations, etc.

The QDP method can been enhanced to perform a variant of the method just mentioned. For instance, a full QDP question might be, "Where was the family car parked in relationship to your childhood home? 1) left side, 2) right side, 3) front, 4) back, 5) under, 6) not close." We can paraphrase the question once the user has learned the numerical answer (by answering it several times), for instance saying just, "Car parked", to which the user can immediately respond with a number. This is a substitution code like SPIN, but it has an advantage that it can be naturally learned.

There are undoubtedly other methods that can be proposed. All will have security and usability issues and tradeoffs. Since it is likely that no single method will be best for all spoken password applications, several methods may be practical.

Acknowledgments. The authors wish to thank Colin Mallows for his invaluable help in working the probabilities of the methods and his insights into their comparative security merits, and Sachin Garg and Navjot Singh for their security analysis and suggestions.

References

1. Haller, N.: The S/KEY One-Time Password System. In: Proc. ISOC Symp. Network and Distributed System Security, San Diego, CA (February 1994)
2. Haller, N., Metz, C., Nesser, P., Straw, M.: A one-time password system. Internet RFC 2289 (1998)
3. Weiss, K.P.: Method and apparatus for positively identifying an individual. U.S. Patent 4720860, January 19 (1988)
4. O'Gorman, L., Bagga, A., Bentley, J.: Call center customer verification by query-directed passwords. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 54–67. Springer, Heidelberg (2004)
5. O'Gorman, L., Bagga, A., Bentley, J.: Query-directed passwords. Computers and Security 24(7), 546–560 (2005)
6. Ellison, C., Hall, C., Milbert, R., Schneier, B.: Protecting secret keys with personal entropy. J. of Future Generation Computer Systems 16(4), 311–318 (2000)
7. Frykholm, N., Juels, A.: Error-tolerant password recovery. In: Samarati, P. (ed.) Eighth ACM Conference on Computer and Communications Security, pp. 1–8. ACM Press, New York (2001)
8. Just, M.: Designing and evaluating challenge-question systems. IEEE Security and Privacy 2(5) (September/October 2004)

9. Dhamija, P., Dhamija, R., Perrig, A.: Déjà Vu: A user study using images for authentication. In: 9th USENIX Security Symposium (2000)
10. Kahn, D.: The Codebreakers, The Story of Secret Writing, Scribner, NY (1996)
11. Bond, M., Danezis, G.: The dining Freemasons (security protocols for secret societies). In: 13th Int. Workshop on Security Protocols, Cambridge, England, April 20-22 (2005)

How to Speak an Authentication Secret Securely from an Eavesdropper

(Transcript of Discussion)

Lawrence O'Gorman

Avaya Labs

Matt Blaze: The model is assuming only one side of the channel will be used liked that?

Reply: Yes, I should reiterate that, because that's very important. The attacker model is, the eavesdropper hears one side, Brutus can attack from the other side, and these guys can collude, but they can't hear both the challenge and the response.

Frank Stajano: What happens if the nurse gets the numbers wrong, does she as many tries as she wants?

Reply: There's a trade-off, but the answer is no, she doesn't get as many tries as she wants.

Ross Anderson: It's worth pointing that if the nurse can only memorise four colours, then for your trust scenario, you're better off giving two digit challenge response rather than four digit challenge response, because you get more entries that way.

Reply: Yes that's interesting, good point.
So we started off with merely four colour substitutions, that was great because the healthcare worker is saying four colour/number substitutions, you know, has a little bit of the load of memorisation, but it's not terrible.

James Malcolm: Couldn't you get the healthcare worker to add the randomisation? It may be not very good, but...

Reply: We could, and we can talk about that.

Matt Blaze: Sorry, maybe I missed something obvious, why do the camouflage elements have to not include the real numbers?

Bruce Christianson: Independence.

Reply: Why do they, because of the histogram attack. There's two scenarios here.

Matt Blaze: Here you never include the elements, but I could imagine a middle ground in which you statistically reduce the incidence of the camouflage elements in the random string of the ones that match the real elements, so that you flatten the histogram, but that still doesn't require you to put everything at the end.

Reply: Don't say anything more, because you're giving away the end of the talk. It took me a long time to figure this out so I'm really mad that you figured it out in a few seconds; I guess everyone else did here, are there any other questions with regards to that?

Bruce Christianson: If you say, we're going to have at least six elements at the end of the colour sequence, and Brutus gets to know you add independence in that way, is it then the case that Brutus gets no information at all?

Reply: Well it depends on the question here. If they get the entire sequence and are allowed to keep on going, then by virtue of the information coming back to them, that, no, this is not authenticated, then they know that something is wrong here. They don't know which colour is wrong, but you're right, there is information that they can get.

Bruce Christianson: Provided Brutus hears at least six numbers after the last colour, he can't tell that he's not in the best case.

Reply: Right, right, now the attacker knows exactly the method that we're using here, and so they'll know this method that, I'll talk about next. We can call this the Blaze solution here, we're going to just do both, adding the dependent camouflage elements which defends against Eve, and then we add independent camouflage elements. Picking a number out of the air, let's try to do this with two thirds probability, that I will have at least three of the four colour/number substitutions in my added camouflage elements. To do that, probabilistically speaking, I have to add ten extra camouflage elements.

Frank Stajano: It looks to me that the main problem you have with these attacks is that Brutus gets to try again and again. You have a very abstract way of framing the problem here where the input from the speaker is just numbers, as if they were punched in a keyboard, instead they are spoken. You obviously need to have a speaker independent number recognition system in there, but you could also use some kind of speaker dependent voice print matching and say, I will behave differently whether I believe that this is Sue, the nurse, or whether it's someone who doesn't sound anything like Sue, maybe because they've got bigger voice, or something like that, so in that case you are going to react differently to whether you let them try again or not.

Reply: I think you're saying two things there but let me comment on one. At the very beginning I said that I'm just throwing biometrics out the window, so we work in a noisy environment, this is a noisy application here and so speaker verification, I'm going to say, doesn't work. I think what you're saying is you can combine it, and maybe this will be a help.

Frank Stajano: Yes, you would be not reliable to do the speaker verification just from the voiceprint, but you use it as a hint to say, how am I going to react when I get a bad match on the digitised links.

Reply: Yes, that's a good comment.

Marios Andreou: How about if the mapping from number to colour is not uniform for every worker, so each person has their own, that's one thought. The other thought is, some of the online banks use a method where when you register you give them the pass phrase, and then they challenge you with three or four random characters from the pass phrase, so they give you 1, 7 and 10 and 12 from the pass phrase. I don't know if that's any more secure than what you've done.

Reply: But it doesn't matter what the scheme is, the eavesdropper is going to hear particular characters each time, and they're going to do this histogram attack.

Bruce Christianson: I assume different employees do have different mappings?

Reply: Oh absolutely, yes.

Tuomas Aura: But in the medical model you show it doesn't help. Having different substitutions for different employees, because an attacker who is listening to only the response, doesn't get any information.

Reply: That's true.

Tuomas Aura: So then, in that attacker model, it doesn't help to have different.

Reply: From the eavesdropper point of view, yes. But if someone gives their password away, you don't want everyone's password to be given away.

Frank Stajano: Explain what memorising voice commands means in this context.

Reply: There are about seven commands that you speak to the voice agent, and you say, connect to Larry O'Gorman, or, connect to the nearest cardiologist, and you can say, get my messages, and, Weaver take a break, Weaver's the name of the voice agent.

Frank Stajano: So when you have an attack, when you discover an attack, you are sure that there is a Brutus in there who's trying things out?

Reply: Well after a certain number of erroneous things you say that, you're guessing.

Frank Stajano: When this happens, what do you do, you have to give them another mapping of colours, surely that's going to annoy the nurses no end?

Reply: Yes. This is obviously the trade-off. There's always trade-offs with security, and, yes, this is one of the things that I'm suggesting. Right now we're very liberal in what we do in the Johns Hopkins test, the security is not high. Well, you know, they turned it off, so it's zero now, but even as it was designed, we didn't ask a whole lot from them.

Frank Stajano: Could you think of a way of, for example, making the mapping not overlap with the previous one, because if you used to have like blue = 7, and green = 9, and then now it's green = 3, and blue = 4, it's certainly going to be mixed up?

Reply: Well that's like a chain hash table, or something, and you can do a chain hash table with a piece of paper.

Frank Stajano: All that I'm suggesting is instead of using colours, the next time you use animals, or shapes, or something cognitively different.

Reply: Well they have to memorise more things, you're asking a person to memorise, to do colour first time, animals second time, and planets.

Frank Stajano: Yes, that's better than colours first time, and different colours the second time, and different colours the third time.

Bruce Christianson: But at least they won't get confused between the second phrase and the first.

Reply: Well that's a good point, yes.

Bruce Christianson: The problem seems to me to be that colours are different from numbers. What if you ask people to remember that certain numbers were not themselves, but the rest were. So four numbers are permuted, all the rest are themselves, and then the challenge is the numbers 0-9 in a random order. That seems to get you both the properties you want.

Reply: Exactly, so the suggestion here is that, I'm giving away information by having a challenge which has numbers and colours in it, because Brutus immediately says, well, the things that aren't colours are numbers, and so I know now what the colour substitution is. So instead of that, having all colours there, but the person knowing that purple is not a colour that they've memorised, the onus is then on the user to say some random number.

Bruce Christianson: I wasn't advocating that, but yes, it could be done.

Ross Anderson: I think what Bruce was suggesting was that you would memorise, for example, 9 mapped to 8, so when the number's 9 you say 8, but 7 maps to nothing, so when 7 is spoken you say a random 2.

Reply: Right, but you don't have to say anything actually.

Ross Anderson: No, you must say a random number.

Bruce Christianson: People are bad at picking random numbers, Eve will get the hint. Instead say there's a permutation of four numbers, all the other numbers map to themselves. Eve will therefore hear the numbers 0 to 9 in a random order, Eve gets no information.

Reply: Yes, so it's just the Eve defence that we still have, and because we have all the numbers there in the challenge we don't have to worry about Brutus anymore.

Ross Anderson: In the threat model where there's one sided of eavesdropping only, that's true. I've another concern with this though, which is completely separate from the threat model. Passwords were great so long as people only ever logged onto the PDP11 in the corner of the lab, but once people had to log onto 101 websites, passwords break badly. Now if your mechanism ever becomes widespread, I suspect it will break badly for the same reason.

Reply: Yes, maybe true.

Bruce Christianson: Or else you have to say that your authentication is always local.

Reply: Right, right, so you're asking not use the same colour/number substitution at computer B, or my second nursing job that I have.

Ross Anderson: Because there's a Mafia controlled part.

Kenny Paterson: If I understand things correctly, users in your system have to produce two kinds of responses. Sometimes they have to repeat a digit, and sometimes they have to translate a colour into a digit? Is there any evidence that users maybe hesitate slightly more when translating?

Reply: That's a good point, and so what we do is, we just jigger the challenges a little bit, so we randomise the time of challenge. But actually the users, when they hear a number have to think, oh that's a number, and they repeat the number, when they hear the colour, they think, oh that's a substitution, and so it takes about the same time in my little experience.

Matt Blaze: So what they're authenticating into is an information system that they're doing independent of their immediate healthcare duties, that is, this isn't, turn on the defibrillator, this is, log me in so that I can get patient records subsequently.

Reply: Yes, I don't know if I'd call that secondary, I mean, they're not speaking to their mum, but, yes, it's not what they're doing with their hands right now.

Matt Blaze: I remember reading Ross' paper about healthcare security, and he pointed out the important issue in healthcare is not confidentiality, but the accountability. And you know, I didn't believe him at all when he first said it, but after thinking about it, I believed that it was obviously true.

Reply: Yes, it's a legal aspect, yes.

Matt Blaze: Finding out who accessed records, is much more important that you preventing someone from getting unauthorised access, because of the emergency access aspects of this. So how do you deal with the problem of an impatient cardiologist wants to find out something right now, and the authentication just doesn't work.

Reply: Well, the answer to that, is always that's a trade-off.

Matt Blaze: Is there a policy built into this system that allows for such breaks, there's something you can do?

Reply: Yes, axe to get through the emergency backdoor.

Matt Blaze: I imagine that's an axe that gets used quite a lot in systems that have annoying properties.

Reply: Well, annoying security properties. This headphone is logged into at the very beginning of the day when hopefully you haven't gotten into your emergency, you wear it all day, you don't have to authenticate each time.

Dan Cvrcek: You could ask them to add some numbers together, instead of repeating the same numbers they are hearing.

Reply: Yes, you could do that, and then that would be random. But then you've asked them to do three things, you've asked to memorise, you've asked them to do the authentication, and you've asked them to do a little bit of maths.

Reply: It's easy for us, you have more confidence in humankind than I do, but that could be the right thing rather than this 20 seconds.

Mike Bond: Alternative technology which might work very well, although it would probably have lower security in terms of total number of permutations, is to use Steganographic channels to do the authentication. George and I did some work on looking at authentication protocols for secret societies, how to design a funny handshake, or a funny walk, that identifies you as a member of a secret society[1]. The way you could apply it here is to have, say, some secret object in the reception, like a vase with flowers, which you put in a different colour every day, and the query to the nurses is, what's the colour, and anybody who listens to that query doesn't know what the object is that you're supposed to say the colour of, but the nurses all know that it's the vase of flowers, or they know that it's the children's toy in the toy rack that's been put on the top of a wardrobe.

Bruce Christianson: That only authenticates whether someone's a nurse or not, it doesn't tell you whether they are the correct one.

Mike Bond: No, but maybe if you start working on that, and you figure out ways to give people different objects, and you figure out ways to alternate the changing of the objects, the nice thing about that is that it uses your associative memory, everybody remembers the key is something in the world around them, and it's something that's really easy to remember, and even quite easy to change. Compared with something abstract, this key is very real, and all you've got to do is the challenge on attributes then.

Frank Stajano: So long as the receptionist doesn't go and rearrange the flowers.

Aaron Coble: I was thinking you have these authentications that are proofs of specific segments in time. If you distributed it out, throughout the day, with just one piece of information being asked, you're working against the eavesdropper

[1] Bond and Danezis, "The Dining Freemasons", 2005, LNCS 4631, pp258-265.

because they no longer know when you're authenticating and when you're not, if you choose your responses carefully.

Reply: Another nice thing about that is it gives you the ability to have different levels of security, logging on, versus going to the morphine cabinet, you know, then you answer another challenge.

Aaron Coble: And a minute and a half sounds like a long time, but if you're being asked one question an hour, it seems. . .

Reply: Yes, then it seems to be a lot less, even though it adds up to a minute and a half, yes.

Secret Public Key Protocols Revisited

Hoon Wei Lim* and Kenneth G. Paterson**

Information Security Group
Royal Holloway, University of London Egham,
Surrey TW20 0EX, UK
{h.lim,kenny.paterson}@rhul.ac.uk

Abstract. Password-based protocols are important and popular means of providing human-to-machine authentication. The concept of secret public keys was proposed more than a decade ago as a means of securing password-based authentication protocols against off-line password guessing attacks, but was later found vulnerable to various attacks. In this paper, we revisit the concept and introduce the notion of identity-based secret public keys. Our new identity-based approach allows secret public keys to be constructed in a very natural way using arbitrary random strings, eliminating the structure found in, for example, RSA or ElGamal keys. We examine identity-based secret public key protocols and give informal security analyses, indicating that they are secure against off-line password guessing and other attacks.

1 Introduction

The use of *secret public keys* in password-based authentication protocols was first proposed by Gong *et al.* [19] in 1993. As implied by its name, a secret public key is a standard public key which can be generated by a user or a server, and is known only to themselves but is kept secret from a third party. A secret public key within a password-based protocol, when encrypted with a user's password, should serve as an unverifiable text[1]. This may significantly increase the difficulty of password guessing even if it is a poorly chosen password as an attacker has no way to verify if he has made the correct guess. The secret public key can then be used by the user for encrypting protocol messages. However, it may not be easy to achieve unverifiability of text by simply performing naive symmetric encryption on public key of standard types such as RSA or ElGamal. This was overlooked in [19] and other variants of secret public key protocols in [18,28], but later found to be the main culprit in various attacks on the protocols. These include undetectable on-line password guessing attacks from

* This author was supported by the EPSRC under grant EP/D051878/1.
** This author was supported by the European Commission under contract IST-2002-507932 (ECRYPT).

[1] Verifiable text/plaintext is a term popularised by Lomas *et al.* in [24]. It refers to a message that contains information that is recognisable when decrypted, whether or not it was predictable in advance.

Ding and Horster [17] and number theoretic attacks due to Patel [25]. It is worth noting that the attacks discovered in [17] may not work against a secret public key protocol which uses a secure public key encryption scheme such as RSA-OAEP [6]. Nevertheless, Patel's attacks seem to be one of the crucial factors that caused diversion of interest away from using secret public keys in password-based protocols. The concept of secret public keys, therefore, was thought to be unworkable. For example, in more recent work on password-based protocols that requires servers' public keys[2] (*e.g.* [11,20]), it is assumed that the public keys are fixed and known to all users.

Contributions. The aims of this paper are twofold: (i) we revisit the notion of secret public keys and uncover some unexplored potential benefits of using identity-based secret public keys, through identity-based cryptography (IBC), in password-based protocols; and (ii) we propose three-party and two-party identity-based secret public key protocols and their respective heuristic security analyses.

In our quest to revive the notion, we introduce some new properties for secret public keys. In the IBC setting, we show that an identity-based secret public key can offer more flexibility in terms of key distribution. For example, an identity-based secret public key can be computed by a user on-the-fly without needing his authentication server to transport the key to him. More importantly, a random string can be used as the identifier for constructing a secret public key. This technique can offer a clean and natural way of eliminating any predictable structure in the secret public key. Through this, the number theoretic attacks that plague existing secret public key protocols can easily be prevented.

Since both public and private keys in the IBC setting are kept secret, we also propose the notion of *secret signatures* which seem to provide data confidentiality in addition to their original cryptographic use, *i.e.* authentication and non-repudiation. This appears to provide additional properties in conventional secret public key protocols and in password-based authentication protocols in general.

Related Work. Extensive work on password-based key exchange protocols (which rely on user passwords only) has already been carried out. See for example [1,2,3,5,13,14,22], which all originate from [8,9]. In order to circumvent off-line password guessing attacks, Bellare *et al.* [5,7] proposed the use of a mask generation function $\mathcal{E}(\cdot)$ as an instantiation of the encryption primitive for encrypting a Diffie-Hellman component, rather than using a standard block (or stream) cipher. For instance, a user with his password PW can encrypt a Diffie-Hellman component g^x by calculating $g^x \cdot H(PW)$, where H is a hash function mapping onto the Diffie-Hellman group and which is modelled as a random oracle in security proofs. Thus the result of the encryption is a group element. This

[2] We classify password-based authentication protocols into two categories: (i) those which require the usage of the server's (or the user's) public key, and sometimes together with the user's password, as a key-encrypted key; and (ii) those which require the user's password only for key transport.

special encryption primitive, which needs to be carefully implemented, is crucial in preventing any information leakage about the password when an attacker mounts a guessing attack. To decrypt and recover g^x, one can simply divide the ciphertext by $H(PW)$. All recent work, such as [1,2,3], utilises this encryption primitive for their password-based key exchange protocols.

The use of algorithms from a public key encryption scheme in a secret key setting is not new. In 1978, Hellman and Pohlig [21] introduced the Pohlig-Hellman symmetric key cipher based on exponentiation. Two different keys are involved in the symmetric key cipher, namely, a secret encrypting key e for the sender and a secret decrypting key d for the receiver, where $e \neq d$. Obviously, the communicating parties must agree in advance to share these two symmetric keys. In more recent work, Brincat [15] investigated how shorter RSA public/private key pairs can be used securely in the secret key world. This is slightly different from [21], as each user has his own secret public/private key pair in [15]. Another related concept is that of public key privacy from Bellare *et al.* [4]. The notion of indistinguishability of keys in public key privacy is an extension of the ciphertext privacy concept: given a set of public keys and a ciphertext generated by using one of the keys, the adversary cannot tell which public key was used to generate the ciphertext. In this chapter, we will make use of identity-based (secret) public keys in the secret key setting. These public keys are known only to the senders and receivers, and thus indistinguishability of encryptions and keys somewhat similar to [4] can be achieved. Moreover, in such a setting, a signature can be made verifiable to only a specific recipient, hence the moniker *secret signature*. In many ways, the concepts of secret public key encryption and signatures seem to be closely related to the notion of signcryption with key privacy from Libert and Quisquater [23]. The proposal of [23] combined Zheng's work on signcryption [30] and the key privacy concept of [4]. Our concept of secret signatures is also related to, but different from, the strongest security notion for undeniable and confirmer signatures called invisibility in [16].

Organisation. The outline of the remainder of this paper is as follows. In Section 2, we review the first proposed secret public key protocol and highlight its problems. Section 3 briefly describes identity-based encryption and signature schemes that will be used in our new approach to secret public keys. In Section 4, we explain and discuss some new properties of secret public keys in the identity-based setting. In Section 5, we propose two variants of identity-based secret public key protocols. We also provide informal security analyses of the protocols. We conclude in Section 6.

2 Secret Public Key Protocols and Attacks

In this section, we revisit the first secret public key protocol proposed in the literature [19]. We will explain what the problems are with the protocol. This will motivate our introduction of identity-based techniques to this area.

Notation. We use $\hat{PK}$ and $\hat{SK}$ to represent a secret public key (SPK henceforth) and its matching private key, respectively. These are no different from conventional asymmetric keys except that they are *both* kept secret. PW denotes a password-derived symmetric key which is shared between a user and an authentication server. A nonce and a random number are represented by n and r, respectively. We use the notation $Enc_{\hat{PK}}(\cdot)$ to indicate asymmetric encryption using a secret public key $\hat{PK}$ and $\{\cdot\}_K$ for symmetric encryption under a symmetric key K. In the three-party scenarios that we will discuss in this section, we use A and B to denote two communicating parties, while S denotes a trusted authentication server whose role is to distribute a copy of a randomly generated session key to both A and B. Other notations will be introduced as they are needed.

The GLNS SPK Protocol. Gong *et al.* [19] envisaged that using secret public keys in a password-based protocol may be useful in a situation where the public keys are needed for certain protocol messages but the protocol participants do not know in advance the public key of their authentication server. In addition, they implicitly assumed that a secret public key could be viewed as a nonce which, when encrypted with a password, offers unverifiability of text. Assuming A and B share their respective passwords with the authentication server S, the server can distribute fresh copies of public keys to A and B encrypted using their respective passwords as symmetric keys at the beginning of each protocol run. Each public key is only known between the server and the relevant participant. This seems to make traditional chosen plaintext attacks more difficult, as the encryption keys are not known to the attacker. The details of the SPK protocol of [19] are depicted in Protocol 1.

Protocol 1. *The GLNS SPK Protocol*

(1). $A \rightarrow S : A,\ B$
(2). $S \rightarrow A : A,\ B,\ n_S,\ \{\hat{PK}_{SA}\}_{PW_A},\ \{\hat{PK}_{SB}\}_{PW_B}$
(3). $A \rightarrow B : Enc_{\hat{PK}_{SA}}(A, B, n_{A1}, n_{A2}, c_A, \{n_S\}_{PW_A}),\ n_S, r_A,\ \{\hat{PK}_{SB}\}_{PW_B}$
(4). $B \rightarrow S : Enc_{\hat{PK}_{SA}}(A, B, n_{A1}, n_{A2}, c_A, \{n_S\}_{PW_A}),$
$\quad Enc_{\hat{PK}_{SB}}(B, A, n_{B1}, n_{B2}, c_B, \{n_S\}_{PW_B})$
(5). $S \rightarrow B : \{n_{A1},\ K_{AB} \oplus n_{A2}\}_{PW_A},\ \{n_{B1},\ K_{AB} \oplus n_{B2}\}_{PW_B}$
(6). $B \rightarrow A : \{n_{A1},\ K_{AB} \oplus n_{A2}\}_{PW_A},\ \{H(r_A),\ r_B\}_{K_{AB}}$
(7). $A \rightarrow B : \{H(r_B)\}_{K_{AB}}$

As shown in Protocol 1, S generates two new sets of secret public/private key pairs $(\hat{PK}_{SA}, \hat{SK}_{SA})$, $(\hat{PK}_{SB}, \hat{SK}_{SB})$ and distributes the public components to A in encrypted form whenever A initiates the protocol run. Here, c_A and c_B are sufficiently large random numbers known as confounders. They serve no purpose other than to confound guessing attacks based on some verifiable texts. Also, H is assumed to be a well-designed hash function.

In [19], the authors assumed that so long as the secret public keys $\hat{PK}_{SA}$ and $\hat{PK}_{SB}$ are randomly generated, it will be difficult for the attacker to verify if his password guesses on $\{\hat{PK}_{SA}\}_{PW_A}$ or $\{\hat{PK}_{SB}\}_{PW_B}$ are correct. In reality, however, this is not completely true. When using conventional public keys such as RSA exponents or Diffie-Hellman components, the keys contain certain number theoretic structure even though they are randomly generated. This, in turn, may allow the attacker to verify his guessed passwords efficiently by predicting and checking the outcome of the decryption. For example, if $\hat{PK}_{SA}$ is an RSA public key of the form $N = pq$, then the attacker could expect the decryption of $\{\hat{PK}_{SA}\}_{PW_A}$ under a guess PW'_A for A's password to be an odd integer. This allows the elimination of half of all passwords in a simple off-line guessing attack. It is this observation that led to Patel's study on various number theoretic attacks on secret public key protocols [25]. It is also worth noting that $\{\cdot\}_K$ must not represent the action of an authenticated encryption algorithm as this would also leak information that could be used to verify the correctness or otherwise of password guesses.

Patel's Attacks. As we have just seen, it can be dangerous to transmit an RSA modulus in encrypted form in an SPK protocol. Even if the ciphertext contains only an RSA exponent, *e.g.* $\{e\}_{PW}$, there are various number theoretic attacks that would reveal the password PW. For example, the attacker could expect the decryption of $\{e\}_{PW}$ under a guess PW'_A to be an odd integer; an even result would eliminate PW'_A as a possible password. Thus, some countermeasures against these number theoretic attacks such as padding or randomisation of the RSA exponent are inevitably required.

Patel [25] showed that even when moduli N are sent in clear, and e are randomised and padded, there is still a lethal off-line guessing attack. Protocol 2 illustrates Patel's RSA version of the SPK protocol. We only show the first 3 out of 7 protocol messages as this is sufficient to describe Patel's attack.

Protocol 2. *The RSA SPK Protocol*

(1). $A \to S : A,\ B$
(2). $S \to A : A,\ B,\ n_S,\ \{e_{SA}\}_{PW_A},\ N_A,\ \{e_{SB}\}_{PW_B},\ N_B$
(3). $A \to B : Enc_{e_{SA}}(A,\ B,\ n_{A1},\ n_{A2},\ c_A,\ \{n_S\}_{PW_A}),\ n_S,\ r_A,\ \{e_{SB}\}_{PW_B}$
$\vdots \qquad\qquad\qquad\qquad \vdots$

An attacker can impersonate S and block A's communication with the real authentication server to mount the following attack.

1. When the attacker E detects A is sending message (1) to S, he blocks S's response from reaching A. E intercepts message (2) and replaces N_A with his own N'_A whose prime factors he knows. Also, since E does not know PW_A, he simply replaces $\{e_{SA}\}_{PW_A}$ with a random string R_A.
2. A unwittingly decrypts R_A with her password-derived key PW_A and obtains e'_{SA} which A believes was generated by S. Subsequently in message (3), A forwards $Enc_{e'_{SA}}(A,\ B,\ \dots)$ to B.
3. E intercepts message (3) and can now perform off-line password guessing on R_A. For each possible PW'_A, E decrypts R_A and retrieves a possible

value for e'_{SA}. Since E knows the prime factors of N'_A, he has no problem computing the decryption exponent d'_{SA} for each value of e'_{SA}. By decrypting $Enc_{e'_{SA}}(A,\ B,\ \ldots)$ with d'_{SA} and checking if the plaintext is of the form $(A,\ B,\ \ldots)$, E can test if PW'_A is the correct password.

It was pointed out in [25] that the above attack on the RSA-based SPK protocol is unavoidable unless all protocol participants use an agreed-upon RSA modulus, or unless the protocol is radically modified.

Even supposing a discrete logarithm based SPK protocol was used, and the ciphertext (which contains a secret public key) transmitted to A was then of the form $\{g^x\}_{PW}$, where g is a generator of a subgroup of $\mathbb{Z}_p^*$ of prime order q and x is a random integer, the password can still be discovered. If a naive encryption of elements in the subgroup is performed with a standard block (or stream) cipher, then there is an off-line password guessing attack. The attacker simply decrypts $\{g^x\}_{PW}$ with a guessed password and observes if the resulting plaintext is an element of the subgroup. If it was an incorrect guess, the likelihood that g^x is not an element of the subgroup is at least $(p-q)/p > 1/2$. This attack can only be prevented by ensuring that decryption of $\{g^x\}_{PW}$ with a guessed password PW' always results in an element of the subgroup. Furthermore, it is also essential that public parameters such as g, p and q have been agreed *a priori* among the users. More examples and discussion on this subject can be found in [25,27]. Notice that this kind of attack is prevented using mask generation functions of the type discussed in Section 1.

From the above descriptions of various number theoretic attacks, it should be evident that designing a SPK protocol can be difficult and not without some extra costs in ensuring the predictable number theoretic structure within public keys is eliminated. These observations are crucial for motivating our identity-based approach. We will show that the aforementioned problems can be prevented easily and naturally, using identity-based techniques.

3 Background on Identity-Based Cryptography

Identity-based cryptography (IBC) was first introduced by Shamir [26]. Recently, there has been an increased intensity in research on IBC. This was mainly due to the seminal discovery of a practical and secure identity-based encryption (IBE) scheme by Boneh and Franklin [10] in 2001. Their scheme uses pairings over elliptic curves.

Background on Pairings. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order q for some large prime q, where $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ denotes a related multiplicative group. A pairing in the context of IBC is a function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties.

– *Bilinear*: Given $P, Q, R \in \mathbb{G}_1$, we have

$$\hat{e}(P, Q + R) = \hat{e}(P, Q) \cdot \hat{e}(P, R) \text{ and } \hat{e}(P + Q, R) = \hat{e}(P, R) \cdot \hat{e}(Q, R).$$

Hence, for any $a, b \in \mathbb{Z}_q^*$, $\hat{e}(aP, bQ) = \hat{e}(abP, Q) = \hat{e}(P, abQ) = \hat{e}(aP, Q)^b = \hat{e}(P, Q)^{ab}$.
- *Non-degenerate*: There exists a $P \in \mathbb{G}_1$ such that $\hat{e}(P, P) \neq 1$.
- *Computable*: If $P, Q \in \mathbb{G}_1$, $\hat{e}(P, Q)$ can be efficiently computed.

For any $a \in \mathbb{Z}_q^*$ and $P \in \mathbb{G}_1$, we write aP as the scalar multiplication of group element P by integer a. Typically, $\mathbb{G}_1$ is obtained as a subgroup of the group of points on a suitable elliptic curve over a finite field, $\mathbb{G}_2$ is obtained from a related finite field, and $\hat{e}$ obtained from the Weil or Tate pairing on the curve.

In what follows, we briefly sketch the popular Boneh and Franklin IBE scheme and an identity-based signature (IBS) scheme with message recovery due to Zhang *et al.* These will be used in our identity-based SPK protocols.

3.1 The Boneh-Franklin Identity-Based Encryption Scheme

The following four algorithms underpin Boneh and Franklin's IBE scheme [10].

Setup**:** Given a security parameter $k \in \mathbb{Z}^+$, the algorithm:
1. specifies two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, and a pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$;
2. chooses an arbitrary generator $P \in \mathbb{G}_1$;
3. defines four cryptographic hash functions, $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$, $H_2 : \mathbb{G}_1^* \to \{0,1\}^n$ for some n, $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$, and $H_4 : \{0,1\}^n \to \{0,1\}^n$; and
4. picks a master secret $s \in \mathbb{Z}_q^*$ at random and computes the matching public component as sP.

The system or public parameters are $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, sP, H_1, H_2, H_3, H_4 \rangle$.

Extract**:** This algorithm is run to extract a private key $sH_1(\text{ID})$ when given an arbitrary identifier string $\text{ID} \in \{0,1\}^*$.

Encrypt**:** To encrypt a message $m \in \{0,1\}^n$ under an identifier ID, the public key used is $Q_{\text{ID}} = H_1(\text{ID})$. The algorithm selects a random $z \in \{0,1\}^n$ and sets $r = H_3(z, m)$. The resulting cipertext is then set to be:

$$c = \langle U, V, W \rangle = \langle rP, z \oplus H_2(g^r), m \oplus H_4(z) \rangle,$$

where $g = \hat{e}(Q_{\text{ID}}, sP) \in \mathbb{G}_2$.

Decrypt**:** To decrypt a ciphertext $c = \langle U, V, W \rangle$ encrypted using the identifier ID, the private key used is $sQ_{\text{ID}} \in \mathbb{G}_1^*$. If $U \notin \mathbb{G}_1^*$, reject the ciphertext. The plaintext m is then recovered by performing the following steps:
1. compute $V \oplus H_2(\hat{e}(sQ_{\text{ID}}, U)) = z$;
2. compute $W \oplus H_4(z) = m$; and
3. set $r = H_3(z, m)$ and if $U \neq rP$, reject the ciphertext, otherwise accept m as the decryption of c.

It is a common assumption that the Setup and Extract algorithms are run by a trusted authority called the Private Key Generator (PKG) within a domain. All users within the domain are assumed to share the same system parameters.

We remark that the above IBE scheme is known to be secure against adaptive chosen ciphertext attacks (IND-ID-CCA) provided the Bilinear Diffie-Hellman problem is hard. This means that even though an adversary has access to some decryption keys associated to some identifiers (apart form the public key ID^* being attacked), he would still not able to deduce any useful information about an encrypted message using ID^* or the decryption key corresponding to ID^*. See [10] for further details.

3.2 The Zhang-Susilo-Mu Identity-Based Signature Scheme with Message Recovery

Using the same notation as above, we describe an IBS scheme with message recovery due to Zhang *et al.* [29].

Setup**:** The PKG selects k_1 and k_2 such that $|q| = k_1 + k_2$. It also defines additional hash functions $H_0 : \{0,1\}^* \to \mathbb{Z}_q^*$, $F_1 : \{0,1\}^{k_2} \to \{0,1\}^{k_1}$, and $F_2 : \{0,1\}^{k_1} \to \{0,1\}^{k_2}$.

The system parameters are now $\langle q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, sP, H_0, H_1, F_1, F_2, k_1, k_2\rangle$.

Extract**:** As above.

Sign**:** Given a private key sQ_{ID} and a message $m \in \{0,1\}^{k_2}$, the signer computes:

1. $v = \hat{e}(P,P)^k$, where $k \in \mathbb{Z}_q^*$;
2. $f = F_1(m) \| (F_2(F_1(m)) \oplus m)$;
3. $r = (H_1(v) + f) \bmod q$; and
4. $U = kP - r(sQ_{\text{ID}})$.

The signature σ is (r, U). The length of the signature is $|r| + |U| = |q| + |\mathbb{G}_1|$.

Verify**:** Given a signature $\sigma = (r, U)$ signed by a user with a public key $Q_{\text{ID}} = H_1(\text{ID})$, the verifier computes

$$f = r - H_1(\hat{e}(U,P)\hat{e}(Q_{\text{ID}}, sP)^r) \text{ and } m = [f]_{k_2} \oplus F_2([f]^{k_1}).$$

The verifier also checks if $[f]^{k_1} = F_1(m)$. The signature is accepted as valid if and only if this equation holds. Here $[f]^{k_1}$ denotes the left-most k_1 bits of the string f, while $[f]_{k_2}$ denotes the right-most k_2 bits of the string f.

The security of this scheme is based on the hardness of the computational Diffie-Hellman problem. To obtain approximately similar security as a standard 1024-bit RSA signature and a 2^{-80} probability of a successful forgery by an adversary, $|k_1| \leq 80$ is needed if a group element of $\mathbb{G}_1$ is represented by 171 bits [29]. The size of the message is limited to k_2, where $k_2 = |q| - k_1$.

4 New Properties from Identity-Based Secret Public Keys

We now present properties from identity-based SPKs by using the Boneh-Franklin IBE and the Zhang-Susilo-Mu IBS schemes. Pre-distribution or fixing of some public/system parameters is common in password-based protocols. In this section and the next, for ease of exposition, we assume that the system parameters for the Boneh-Franklin IBE and the Zhang-Susilo-Mu IBS schemes can be distributed by the server to all its users during the user registration phase using an out-of-band mechanism. This is important as failure to use an authentic set of system parameters would allow the attacker to inject his own chosen parameters. Also, during the registration phase between a user and the server, the user will pick a password pwd and send an image PW of the password to the server. Typically, one might set $PW = H_0(pwd) \cdot P$, where $H_0 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $\mathbb{G}_1$ is a group of prime order q used elsewhere in the protocol, and P generates $\mathbb{G}_1$. Note then that the server only knows PW and not pwd. The actual password pwd still remains private to the user only. In some cases where pwd and PW are used together, stronger authentication can be provided in the sense that the user's authenticity can still be guaranteed even if the string PW stored in the server is revealed. This technique of using an image of the actual user-selected password is common to many password-based protocols, for example [1,5,7,9].

Here, we present and discuss some interesting properties of identity-based SPKs (ID-SPKs henceforth) which are new as compared to conventional SPK protocols based on RSA or Diffie-Hellman. These properties can be obtained from using the Boneh-Franklin and Zhang-Susilo-Mu schemes, and they form the basis and motivation for the ID-SPK protocols that we will discuss in Section 5.

4.1 ID-SPK as Secret Identifier

In the conventional IBC setting, an identifier refers to some public information which represents a user and is known to all parties. Here, however, we work with secret identifiers, that is, identifiers only known to the user A (or B) and the server S. These can be obtained by binding a secret value such as a password to an identifier. Such an ID-SPK of the form $\hat{PK} = H_1(\texttt{user} \,\|\, \texttt{password} \,\|\, \texttt{policy})$ can be generated by both the user and the server on-the-fly. Here `policy` denotes constraints that can be included in the ID-SPK such as a date, nonces, or roles. In other words, the server does not need to distribute a fresh secret public key to its users, in contrast to [19,28]. Here we assume the users have access to the server's fixed system parameters. For example, referring back to Protocol 1, when A initiates the protocol she could, in principle, skip messages (1) & (2) and transmit message (3) to B as follows:

$$(3).\ A \rightarrow B : Enc_{\hat{PK}_{AS}}(A, B, \ldots)$$

where $\hat{PK}_{AS} = H_1(A \| B \| S \| PW_A \| \text{“10102005”})$ denotes a public key in the IBE scheme of [10]. Here “10102005” represents a date. A date with more granularity (*e.g.* concatenated with time) or a nonce may well be needed to ensure freshness of $\hat{PK}_{AS}$. We remark that the Boneh-Franklin IBE scheme is probabilistic and thus the attacker cannot use a guessed password PW'_A to verify his guess by generating $Enc_{\hat{PK}_{AS}}(A, B, \ldots)$ and comparing it with the actual ciphertext produced by A, even if he knows all the plaintext components.

On the server side, the server can extract the matching private key for $\hat{PK}_{AS}$ using its master secret. Unless the attacker can break the IBE scheme or recover the master secret, the above ciphertext is resistant to password guessing attacks. This identity-based technique offers a form of non-interactive distribution of secret public keys from the server to its users.

In the above example, A uses an ID-SPK encryption scheme which is adapted from the full version of the IBE scheme of [10] with the encryption key only known to the user and the server. Formal security definitions and proofs of security for ID-SPK encryption schemes are beyond the scope of this paper and will be addressed in our future work.

4.2 Random String as ID-SPK

We have explained earlier in Section 2 that a naive encyption of an RSA exponent or a group element with a standard block cipher would lead to effective off-line password guessing attacks. Therefore, some form of padding or randomisation of the keys is needed. In the IBC setting, we note that a random string with arbitrary length without any predictable structure can also be used as an identifier. The corresponding public key can be derived by hashing. Since now only a random string needs to be encrypted under the user password, the possibility of using a standard block cipher for the encryption is opened up[3]. For example, in Protocol 1, the server can transport random strings ST_A and ST_B to A and B, respectively, in message (2) as follows:

(2). $S \rightarrow A: A,\ B,\ n_S,\ \{ST_A\}_{PW_A},\ \{ST_B\}_{PW_B}$
(3). $A \rightarrow B: Enc_{\hat{PK}_{SA}}(A,\ B,\ n_{A1},\ n_{A2},\ n_S),\ n_S,\ r_A,\ \{ST_B\}_{PW_B}$
(4). $B \rightarrow S: Enc_{\hat{PK}_{SA}}(A,\ B,\ n_{A1},\ n_{A2},\ n_S), Enc_{\hat{PK}_{SB}}(B,\ A,\ n_{B1},\ n_{B2},\ n_S)$

Since ST_A and ST_B are just random strings, they do not contain any predictable structure which may leak some information to the attacker as in the case of RSA or Diffie-Hellman keys. Subsequently, users A and B can derive their ID-SPKs $\hat{PK}_{SA} = H_1(A\|B\|S\|ST_A)$ and $\hat{PK}_{SB} = H_1(B\|A\|S\|ST_B)$, respectively, and respond to the server via messages (3) and (4). If the server can decrypt B's reply and recover n_S from both the ciphertexts produced with $\hat{PK}_{SA}$ and $\hat{PK}_{SB}$, it can be assured that the users have received the correct random strings. Thus, A and B are authenticated to S. The use of random strings as identifiers is a key property from our identity-based approach which may give the concept of SPK protocols new life.

We remark that to prevent off-line attacks, ciphertexts obtained by encryption under the keys $\hat{PK}_{SA}$ and $\hat{PK}_{SB}$ must not leak useful information about ST_A and ST_B, respectively. This is not a traditional requirement of a public key encryption scheme (it is related to the public key privacy concept in [4]). Also note that since we use a probabilistic encryption scheme here, we have removed the use of confounders c_A and c_B originally proposed in Protocol 1 in messages

[3] However, it is still necessary to take care to avoid attacks based on the introduction of redundancy, for example padding, in the block cipher encryption.

(3) and (4). Furthermore, users A and B no longer need to encrypt n_S with their respective passwords in their replies to S, in messages (3) and (4). This is because users A and B can demonstrate their knowledge of respective passwords by their ability to construct correct keys from ST_A and ST_B.

4.3 Secret Signatures

In what follows, we show some extended properties that an ID-SPK can offer as compared to a conventional SPK. Again, referring to Protocol 1, if in the protocol A (and B) selects and sends ST_A (and ST_B) to the server (rather than the server sending it to the user), we can, in principle, remove messages (1) & (2) and modify messages (3) – (5) as follows:

$$\begin{aligned}
&(3).\ A \to B : Enc_{\hat{PK}_{SA1}}(A,\, B,\, ST_A),\, r_A\\
&(4).\ B \to S : Enc_{\hat{PK}_{SA1}}(A,\, B,\, ST_A),\, r_A,\\
&\qquad\qquad\quad Enc_{\hat{PK}_{SB1}}(A,\, B,\, ST_B),\, r_B\\
&(5).\ S \to B : Sig_{\hat{SK}_{SA2}}(K_{AB}),\, Sig_{\hat{SK}_{SB2}}(K_{AB})
\end{aligned}$$

Note that we have replaced nonces n_{A1}, n_{A2}, n_{B1} and n_{B2} in Protocol 1 by random strings ST_A and ST_B. For ease of exposition, we concentrate on the interaction between A and S. In message (3), A encrypts a random string ST_A with an ID-SPK $\hat{PK}_{SA1} = H_1(A\|B\|S\|PW_A)$. It is obvious that a symmetric encryption of the form $\{A,\, B, \ldots,\, ST_A\}_{PW_A}$ cannot be used in message (3) because the identities of A and B are verifiable texts. The server responds with a signature generated with a private key associated with the public key $\hat{PK}_{SA2} = H_1(S\|A\|B\|PW_A\|ST_A)$. The reason for doing this will be clear when we look at the motivation for using $Sig_{\hat{SK}}(\cdot)$, a signature scheme with a private key $\hat{SK}$, in message (5). As compared to the modification of Protocol 1 given in Section 4.2, the server cannot reply to A with an encrypted message using an ID-SPK constructed from $H_1(S\|A\|B\|PW_A\|ST_A)$. This is mainly because in such an asymmetric model (where the user only knows an easy-to-remember password and the server has access to the secret public/private key pairs), only the server itself can extract the corresponding private key. This prompts the requirement to use a secret signature which not only provides non-repudiation of the signed message and message recovery, but also preserves message confidentiality. This last property is needed because the server wants only A and B to be able to verify the signatures and recover the signed messages. This, in turn, leads us to the use of an ID-SPK signature scheme with message recovery which can be adapted from [29]. So long as the verification keys used in the scheme of [29] are kept secret between the intended parties, our concept of secret signatures can be used. However, we remark that the IBS scheme with message recovery must be used carefully because the scheme provides message integrity. In other words, a simple off-line password guessing attack would be enabled if a secret signature was created based on a private key corresponding to $\hat{PK}_{SA2} = H_1(S\|A\|B\|PW_A)$. For instance, the attacker could construct an ID-SPK $\hat{PK'}_{SA2} = H_1(S\|A\|B\|PW'_A)$ using a guessed password PW'_A and then attempt to verify the signature. If he used the wrong password, the Verify algorithm would return an error message. Because of that, the identifier from which the verifying key is derived must contain a secret value chosen from a space much larger than the password space. We achieve this by including ST_A (or ST_B) in the identifier. It is also worth mentioning that a secret signature should not leak information about the signing key, the verifying key, or the plaintext that has been signed.

As we have explained earlier, a secret identifier can bind a user's password naturally to a secret public/private key pair. As such, secret signatures may be beneficial in a password-based protocol when one or both of the following conditions apply:

(i). Non-repudiation, confidentiality, and integrity of a signed message are required.
(ii). An additional line of defence is desirable (*e.g.* assuming the server keeps its master secret in a tamper-resistant hardware token or smartcard, the attacker cannot impersonate the server to any of its users even if the users' passwords are exposed).

Security definitions and proofs of security for ID-SPK signature schemes with message recovery will be addressed in our subsequent work on secret public keys.

5 The ID-SPK Protocols

In the previous sections, we learned that to exploit the advantages of using SPKs in a password-based protocol, the keys must not contain any predictable structure, such as that appearing in RSA or discrete logarithm-based systems. This section presents complete three-party and two-party ID-SPK protocols which can solve this structural issue in a clean and natural way. These protocols build on the ideas introduced in the previous section. We assume that all the protocol participants have agreed on some public/system parameters for the ID-SPK encryption and signature schemes *a priori.*

Before we look at the ID-SPK protocols, it may be useful to classify some common attacks on password-based protocols.

– *On-line password guessing attacks*: The attacker chooses a password from his dictionary and tries to impersonate a user. He verifies the correctness of his guess based on responses from the server. If the impersonation fails, the attacker tries again using a different password from his dictionary. Note that the attacker can also impersonate the server to the user by intercepting and modifying a message originating from the server before forwarding it to the user (assuming the server has used the user's password in some way in creating the message). He can then verify his password guesses based on responses from the user.
– *Off-line password guessing attacks*: The attacker records past communication and makes a verifiable guess using a password from his dictionary. If the

guess fails, the attacker tries again with a different password until the correct password is found. No on-line participation of a server (or a user) is required and the attacks take place without the knowledge of the actual protocol participants.

– *Attacks exploiting exposed secrets*: The attacker may occasionally have access to sensitive information such as past session keys or a user's password. This is possible when the user's machine or the server are compromised, or the user's password is revealed through a keystroke logger. It is a desirable security property that exposure of past session keys will not lead to the exposure of the user's password and vice versa.
– *Undetectable on-line password guessing attacks*: The attacker mounts an on-line guessing attack. However, a failed guess cannot be detected and logged by the server (or the user). In other words, the protocol participants cannot distinguish a genuine protocol message from a modified (malicious) message.

Security Model. We sketch here our definition of the security for a password-based ID-SPK protocol, using an informal security model. In the model, there is an adversary E, who is allowed to watch regular runs of the protocol between a user, $U \in \mathcal{U}$, where $\mathcal{U}$ is a set of protocol users, and a server S. E can actively communicate with the user and the server in replay, impersonation, and man-in-the-middle attacks. The adversary can prompt one of the parties to initiate new sessions. In each session, E can see all the messages sent between U and S. Furthermore, he can intercept the messages and modify or delete them. Also, E gets to see whether S accepts the authentication or not. In addition, we allow the adversary to establish as many "accounts" as he wishes with the server using his own chosen passwords. He can then run arbitrarily many authentication sessions using these accounts to obtain information for his attacks.

It is clear that if the user picks a password from his dictionary $\mathcal{D}$, then the adversary that attempts n active impersonation attacks (or on-line guessing attacks) over n distinct sessions with the server can succeed with probability at least $n/|\mathcal{D}|$ by trying a different password from $\mathcal{D}$ in each attempt.

Definition 1 (*Informal*). *We say that the ID-SPK protocol is secure if all the following conditions are satisfied.*

1. *No useful information about a session key is revealed to the adversary during a successful protocol run and the exposure of past session keys does not leak any information about the current session key.*
2. *The adversary cannot discover the correct user password after n active impersonation attempts with probability significantly higher than $n/|\mathcal{D}|$.*
3. *The protocol is resistant to off-line password guessing attacks.*
4. *The protocol is resistant to undetectable on-line password guessing attacks.*
5. *The exposure of the user's past session keys will not lead to the exposure of the user's password and vice versa.*

We remark that formal security model and definition, such as those used in [5], have not been employed in this paper. This is because the main objective of the paper is to explore new ways of using SPKs in the IBC setting.

5.1 The Three-Party ID-SPK Protocol

In [18], Gong further optimised the original SPK protocol in [19] by reducing the number of protocol messages to reduce the communication costs incurred by the protocol. We further modify Gong's optimised SPK protocol by building on the example given in Section 4.3, as shown in Protocol 3.

Protocol 3. *The Modified Gong SPK Protocol*

(1). $A \rightarrow B : A,\ r_A,\ Enc_{\hat{PK}_{A1}}(A,\ ST_A)$
(2). $B \rightarrow S : B,\ Enc_{\hat{PK}_{B1}}(B,\ ST_B),\ A,\ r_A,\ Enc_{\hat{PK}_{A1}}(A,\ ST_A)$
(3). $S \rightarrow B : Sig_{\hat{SK}_{B2}}(K_{AB}),\ Sig_{\hat{SK}_{A2}}(K_{AB})$
(4). $B \rightarrow A : Sig_{\hat{SK}_{A2}}(K_{AB}),\ \mathrm{MAC}_{F(K_{AB})}(B,\ A,\ r_A),\ r_B$
(5). $A \rightarrow B : \mathrm{MAC}_{F(K_{AB})}(A,\ B,\ r_B)$

In Protocol 3, users A and B select their respective random strings ST_A and ST_B and encrypt them with an ID-SPK. As before, $\hat{PK}_{A1} = H_1(A\|B\|S\|PW_A)$ and $\hat{PK}_{B1} = H_1(B\|A\|S\|PW_B)$. The server recovers ST_A and ST_B, and computes private keys $\hat{SK}_A$ and $\hat{SK}_B$ matching the ID-SPKs $\hat{PK}_{A2}$ and $\hat{PK}_{B2}$ constructed from the random strings: $\hat{PK}_{A2} = H_1(S\|A\|B\|PW_A\|ST_A)$ and $\hat{PK}_{B2} = H_1(S\|B\|A\|PW_B\|ST_B)$. The private keys are then used to sign a session key. We assume that an IBS scheme with message recovery is used, so that the intended recipients are able to recover the session key using their knowledge of the ID-SPKs. These secret signatures also provide non-repudiation. Even though this is rarely a requirement in protocols for authentication and key establishment, it automatically provides the important data integrity and data origin authentication services [12]. Note that $\mathrm{MAC}_{F(K_{AB})}(B,\ A,\ r_A)$ and $\mathrm{MAC}_{F(K_{AB})}(A,\ B,\ r_B)$ in messages (4) and (5) are used by A and B, respectively, to prove to each other that they are indeed sharing the same session key. This provides key confirmation. Here, F denotes a key derivation function.

To improve the performance of Protocol 3, $Sig_{\hat{SK}_{A2}}(K_{AB})$ and $Sig_{\hat{SK}_{B2}}(K_{AB})$ in message (3) can be replaced with $\{K_{AB}\}_{F(ST_A)}$ and $\{K_{AB}\}_{F(ST_B)}$, respectively.

Security Analysis. Protocol 3 shows that users A and B communicate with S using secret identifiers $\mathrm{ID}_A = A\|B\|S\|PW_A$ and $\mathrm{ID}_B = B\|A\|S\|PW_B$, respectively. These identifiers involve the users' passwords. Since S is the only party who has knowledge of PW_A and PW_B apart from A and B, the users should receive the same session key created by the server provided the correct private keys are used to transport the session key. If A and B can successfully recover K_{AB} from their respective received secret signatures, they can be assured of the authenticity of the server.

It is clear that requirement 1 of Definition 1 can be satisfied if the session key is randomly generated by the server. Moreover, the session key cannot be computed directly by the adversary E.

By observing a protocol run, E can gather information by intercepting the protocol messages, such as $Enc_{\hat{PK}_{A1}}(A,\ ST_A)$, $Enc_{\hat{PK}_{B1}}(B,\ ST_B)$, $Sig_{\hat{SK}_{A2}}(K_{AB})$

and $Sig_{\hat{SK}_{B2}}(K_{AB})$. However, since we assume that the ID-SPK encryption scheme used in this protocol is IND-ID-CCA secure, E cannot gain any useful information about ST_A and ST_B from $Enc_{\hat{PK}_{A1}}(A, ST_A)$ and $Enc_{\hat{PK}_{B1}}(B, ST_B)$ without knowledge of the master secret held by the server. As for the session key transportation in the form of secret signatures from the server to the users, E can choose his own verification keys in an attempt to recover the session key. However, there seems to be no efficient way for E to predict the correct ID-SPK if the ID-SPK signature scheme used in the protocol offers appropriate security. In particular, we assume that E cannot distinguish a secret signature from a randomly generated string if the identifier is constructed using sufficient randomness. We also assume that the adversary cannot forge valid secret signatures, impersonating the server to users. Apart from that, it is very unlikely that E can impersonate a legitimate user by guessing the user's password. This is so since the adversary's impersonation attack would be detected immediately by the server if the user's chosen random string cannot be recovered successfully from message (2). Note that the number of impersonation attempts can be kept acceptably small by using mechanisms that can log and control the number of failed authentication attempts. A brute force attack on message (3) or (4) to deduce the session key can be easily thwarted by using random strings ST_A and ST_B with entropy significantly larger than the password space of $\mathcal{D}$. Also, so long as ST_A and ST_B are fresh and randomly generated for each protocol run, E would not be able to mount a replay attack. It is thus conjectured that requirement 2 is satisfied.

When E uses a password $PW'_A \in \mathcal{D}$ to mount an off-line password guessing attack on a recorded $Enc_{\hat{PK}_{A1}}(A,\ ST_A)$, there is no way for the adversary to verify the correctness of $\hat{PK'}_{A1} = H_1(A\|B\|S\|PW'_A)$ if the ID-SPK encryption is randomised and IND-ID-CCA secure. If E selects $PW'_A \in \mathcal{D}$ and ST'_A at random, computes $\hat{PK'}_{A2} = H_1(S\|A\|B\|PW'_A\|ST'_A)$, and then attempts to verify $Sig_{\hat{SK}_{A2}}(K_{AB})$, his check will almost certainly fail since the entropy of ST_A is much larger than the entropy of PW_A. Thus this form of off-line guessing attack will not succeed and therefore, Protocol 3 also satisfies requirement 3.

If E has a valid account with S, he may possibly mount an insider attack by impersonating A to S, pretending to be wanting to establish a session key K_{AE} with himself. In the attack, E initiates the protocol by computing $Enc_{\hat{PK'}_{A1}}(A,\ ST'_A)$ with a guessed password PW'_A, and hence $\hat{PK'}_{A1} = H_1(A\|B\|S\|PW'_A)$. However, once this message has reached S, the server should get an error message when decrypting $Enc_{\hat{PK'}_{A1}}(A, ST'_A)$ using the decryption key matching $\hat{PK'}_{A1}$. Therefore it is clear that the protocol can detect on-line guessing attacks and thus requirement 4 is satisfied.

On certain rare occasions, E may have access to A's or B's machine and thus the past session keys shared between them are exposed. However, since E has no knowledge of the master secret of S and the matching private component of $\hat{PK}_{A2}$, E still cannot determine PW_A even though he can mount a brute-force attack on $\hat{PK}_{A2}$. On the other hand, if for some reason, E has the correct password for A, he may attempt to find the value of ST_A given A's password and the ciphertext $Enc_{\hat{PK}_{A1}}(A,\, ST_A)$. Since the encryption scheme is IND-ID-CCA secure, E only has a negligible success probability to discover the correct ST_A. Also, since the value of the verification key for $Sig_{\hat{SK}_{A2}}(K_{AB})$ depends on the secret value ST_A, E can only recover the session key with negligible probability and forward secrecy of the protocol is preserved. Hence, requirement 5 is also satisfied. We conclude that Protocol 3 is a secure ID-SPK assuming that the ID-SPK encryption and signature schemes are appropriately secure.

Nevertheless, it is worth noting that if the server's master secret is compromised, the adversary can deduce the users' passwords without much difficulty. For instance, for each candidate password PW'_A, E can extract the private key matching the identifier $\mathrm{ID}'_A = A \| B \| S \| PW'_A$ and use it to attempt to decrypt $Enc_{\hat{PK}_{A1}}(A,\, ST_A)$ from message (1), and check if the decryption unveils A's identity. Hence, it is of the utmost importance that the server's master secret is kept private, for example by using a strong protective mechanism such as storing it in a tamper-resistant device.

5.2 The Two-Party ID-SPK Protocol

We now present a Diffie-Hellman type two-party ID-SPK protocol. Our protocol is adapted from [1,5] which make use of an encrypted Diffie-Hellman ephemeral key exchange. We apply the identity-based techniques that we introduced in Section 4 to obtain Protocol 4, as shown below.

Protocol 4. *The Diffie-Hellman ID-SPK Protocol*

$$\begin{aligned}
&(1).\ A \rightarrow S : A,\ Enc_{\hat{PK}_{A1}}(aP)\\
&(2).\ S \rightarrow A : S,\ Sig_{\hat{SK}_{A2}}(xP)
\end{aligned}$$

In Protocol 4, the user randomly selects $a \in \mathbb{Z}_q^*$ and computes aP, where $P \in \mathbb{G}_1$ is part of the system parameters. A then encrypts the Diffie-Hellman component with $\hat{PK}_{A1} = H_1(A \| S \| PW_A)$ and sends message (1) to S. The server extracts the matching private key $\hat{SK}_{A1}$ with its master secret to recover aP. Subsequently, S picks a random number $x \in \mathbb{Z}_q^*$ and calculates xP. The server then extracts another private key which is associated with $\hat{PK}_{A2} = H_1(S \| A \| PW_A \| aP)$, produces $Sig_{\hat{SK}_{A2}}(xP)$, and transmits it to A. After receiving message (2), the user retrieves xP with $\hat{PK}_{A2}$. Both the user and the server calculate a session key as $K_{AS} = F(A \| S \| PW_A \| aP \| xP \| axP)$, where F is a key derivation function. Note that key confirmation can be provided by adding a third message from A to S, in which A provides a MAC computed on all the protocol messages using the session key (derived using a different key derivation function to F).

Security Analysis. As with Protocol 3, user A uses an ID-SPK, but in this case to transport a Diffie-Hellman ephemeral key aP to the server. It is worth noting that message (1) can be replayed but this is not an issue because the purpose of the protocol is to authenticate the session key. If the adversary E

has captured message (1) and replays it, he will not gain any information about the session key, unless he has access to a and to xP in message (2). Also, we note that since only S other than A has access to PW_A, S is authenticated to A when A successfully recovers xP using $\hat{PK}_{A2} = H_1(S\|A\|PW_A\|aP)$ (recall that an ID-SPK signature scheme provides a message integrity check).

Clearly, requirement 1 of Definition 1 can be satisfied if the ephemeral Diffie-Hellman components from A and S are randomly generated and information used to compute the session key including a, aP, x, xP, and PW_A cannot be computed directly by E.

E has access to $Enc_{\hat{PK}_{A1}}(aP)$ and $Sig_{\hat{SK}_{A2}}(xP)$ through watching a protocol run between A and S. However, since we assume that the ID-SPK encryption scheme used in this protocol is IND-ID-CCA secure, E cannot obtain any useful information about aP from $Enc_{\hat{PK}_{A1}}(aP)$ without knowledge of the master secret held by the server. Also, we assume that the ID-SPK signature scheme used in the protocol produces secret signatures $Sig_{\hat{SK}_{A2}}(xP)$ that are indistinguishable from random strings. Hence it is hard for E to deduce any information about the Diffie-Hellman component chosen by the server. Using analysis similar to that we used when discussing Protocol 3, it appears unlikely that E will successfully impersonate A in n attempts with probability significantly higher than $n/|\mathcal{D}|$ or mount a replay attack, provided aP and xP are fresh and their entropy is significantly higher than the entropy of $\mathcal{D}$. Also, the use of an incorrect password in generating $\hat{PK}_{A1}$ can be easily detected by the server when the server uses the wrong matching private key to recover aP. It is thus conjectured that requirements 2, 3 and 4 are satisfied.

It is possible that E may have access to A's machine and recover the past session keys used by A. In that case, despite the fact that E knows K'_{AS}, he must be able to reverse the key derivation function F in order to deduce A's password. On the other hand, if for some reason A's password is revealed to E, E may attempt to find the value of aP given A's password and the ciphertext $Enc_{\hat{PK}_{A1}}(aP)$. Since the encryption scheme is IND-ID-CCA secure, E only has a negligible success probability to find the correct aP. Also, since the value of the verification key for $Sig_{\hat{SK}_{A2}}(xP)$ depends on the secret value aP, E can only recover the session key with negligible probability. This is related to the forward secrecy of protocols discussed in [1,5]. Therefore, requirement 5 is also satisfied. We note that in addition to having met this requirement, even if E knows aP and xP, he has to solve the intractable CDH problem in order to calculate axP and hence the session key. We conclude that Protocol 4 is a secure ID-SPK protocol.

As with the security of Protocol 3, it is essential to have the server's master secret adequately protected to ensure that the aforementioned security conditions hold.

6 Conclusions

We studied the history of secret public key protocols and discussed some known problems with these protocols. We then explored some interesting properties of identity-based cryptography which form the basis of our proposed identity-based secret public key protocols. These properties also allow us to convert a conventional identity-based encryption scheme and a standard identity-based signature scheme (with message recovery) into their secret public key equivalents.

We presented three-party and two-party identity-based secret public key protocols for key exchange. Our heuristic security analyses show that the protocols appear to be secure against off-line password guessing attacks and undetectable on-line password guessing attacks, and provide forward secrecy. The security definitions and proofs of the ID-SPK encryption and signature schemes, as well as formal security analyses of the proposed ID-SPK protocols in this paper, will be addressed in our further work on this subject.

Acknowledgements

The authors would like to thank Qiang Tang for his very helpful feedback on an earlier draft of this paper, and Kim-Kwang Raymond Choo for pointing out a flaw in an earlier version of Protocol 3.

References

1. Abdalla, M., Chevassut, O., Pointcheval, D.: One-time verifier-based encrypted key exchange. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 47–64. Springer, Heidelberg (2005)
2. Abdalla, M., Fouque, P., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
3. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005)
4. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
6. Bellare, M., Rogaway, P.: Optimal asymmetric encryption – how to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995)
7. Bellare, M., Rogaway, P.: The AuthA Protocol for Password-Based Authenticated Key Exchange. Contribution to IEEE P1363 (March 2000)
8. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: Proceedings of the 1992 IEEE Symposium on Security and Privacy, pp. 72–84. IEEE Computer Society Press, Los Alamitos (1992)
9. Bellovin, S.M., Merritt, M.: Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In: Proceedings of the 1st ACM Computer and Communications Security Conference, pp. 244–250. ACM Press, New York (1993)

10. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
11. Boyarsky, M.K.: Public-key cryptography and password protocols: The multi-user case. In: Proceedings of the 6th ACM Computer and Communications Security Conference, pp. 63–72. ACM Press, New York (1999)
12. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer, Berlin (2003)
13. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password authenticated key exchange using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000)
14. Bresson, E., Chevassut, O., Pointcheval, D.: Security proofs for an efficient password-based key exchange. In: Proceedings of the 10th ACM Computer and Communications Security Conference, pp. 241–250. ACM Press, New York (2003)
15. Brincat, K.: On the use of RSA as a secret key cryptosystem. Designs, Codes, and Cryptography 22(3), 317–329 (2001)
16. Chaum, D., van Heijst, E., Pfitzmann, B.: Cryptographically strong undeniable signatures, unconditionally secure for the signer. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 470–484. Springer, Heidelberg (1992)
17. Ding, Y., Horster, P.: Undetectable on-line password guessing attacks. ACM Operating Systems Review 29(4), 77–86 (1995)
18. Gong, L.: Optimal authentication protocols resistant to password guessing attacks. In: Proceedings of 8th IEEE Computer Security Foundations Workshop (CSFW 1995), pp. 24–29. IEEE Computer Society Press, Los Alamitos (1995)
19. Gong, L., Lomas, T.M.A., Needham, R.M., Saltzer, J.H.: Protecting poorly chosen secrets from guessing attacks. IEEE Journal on Selected Areas in Communications 11(5), 648–656 (1993)
20. Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. ACM Transactions on Information and System Security 2(3), 230–268 (1999)
21. Hellman, M.E., Pohlig, S.C.: Exponentiation Cryptographic Apparatus and Method. U.S. Patent #4,424,414, January 3 (1984) (expired)
22. Jablon, D.P.: Strong password-only authenticated key exchange. ACM SIGCOMM Computer Communication Review 26(5), 5–26 (1996)
23. Libert, B., Quisquater, J.-J.: Efficient signcryption with key privacy from gap Diffie-Hellman groups. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 187–200. Springer, Heidelberg (2004)
24. Lomas, T.M.A., Gong, L., Saltzer, J.H., Needham, R.M.: Reducing risks from poorly chosen keys. ACM Operating Systems Review 23(5), 14–18 (1989)
25. Patel, S.: Number theoretic attacks on secure password schemes. In: Proceedings of the 1997 IEEE Symposium on Security and Privacy, pp. 236–247. IEEE Computer Society Press, Los Alamitos (1997)
26. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
27. Steiner, M., Buhler, P., Eirich, T., Waidner, M.: Secure password-based cipher suite for TLS. ACM Transactions on Information and System Security 4(2), 134–157 (2001)
28. Tsudik, G., Herreweghen, E.V.: Some remarks on protecting weak keys and poorly chosen secrets from guessing attacks. In: Proceedings of the 12th IEEE Symposium on Reliable Distributed Systems (SRDS 1993), pp. 136–141. IEEE Computer Society Press, Los Alamitos (1993)

29. Zhang, F., Susilo, W., Mu, Y.: Identity-based partial message recovery signatures (or how to shorten ID-based signatures). In: Patrick, A.S., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 45–56. Springer, Heidelberg (2005)
30. Zheng, Y.: Digital signcryption or how to achieve cost (Signature & encryption) << cost(Signature) + cost(Encryption). In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 165–179. Springer, Heidelberg (1997)

Secret Public Key Protocols Revisited
(Transcript of Discussion)

Hoon Wei Lim

Royal Holloway, University of London

The concept of using secret public keys in designing security protocols is not new. It was first proposed more than ten years ago, and today we're going to revisit the concept, discuss a problem with the concept, and propose some fixes with identity-based cryptography.

Bruce Christianson: I'm sorry to interrupt, but can you just remind us why you can't, for example, encrypt an El Gamal key by adding the password modulo the base?

Reply: Yes, just exactly what I'm going to say next. Standard block ciphers cannot be used to encrypt standard public keys, not without some form of randomisation or padding. Or alternatively, in recent years, there were proposals of using a special mask generation function as the encryption primitive, but the function works in a way that every decryption of an encrypted public key will give you the same number theoretic structure as before it was encrypted, so that is another approach.

In the standard identity-based setting, an identifier is just some public information, and it's usually assumed that this information is known to all parties, but here we work with secret identifiers, which are only known between the users and the server. One advantage of this is that a user can compute an identity-based secret public key on the fly without having the server to generate a new secret public key, and deliver the key to the user.

Remember that a naive symmetric encryption of a standard public key will lead to various effective guessing attacks.

Bruce Christianson: The public keys have to remain secret?

Reply: Yes, exactly, so it's known between the users and the server.

Bruce Christianson: Any break is retrospective? If I ever learn the public key, then I learn enough to search on the password?

Reply: Yes, exactly. So we observed that a random string with arbitrary length, and without any predictable structure, can also be used as an identifier, and now what needs to be encrypted is just a random string, so the possibility of using a standard block cipher is back. For example, the server can generate a random string ST_A and encrypt the string with the user password, so user A recovers ST_A, and uses it to compute a new secret public key. The corresponding private key can only be computed by the server, because the server knows the master secret. Now the guessing attack doesn't work here because any choice of passwords would lead to a varied choice of string ST_A.

Here's another property of identity-based secret public keys. We are working in an asymmetric model where the server knows the master secret which it can use to extract private keys, and on the other hand, the user only knows the password, and in addition the user can produce identity-based secret public keys, based on some shared secret. So the server cannot encrypt a message using identity-based secret public keys because the user will not be able to decrypt the message, but on the other hand, the server can produce a signature which can only be verified by the user. This is because the server can extract a private key for signing of which the verification key can only be computed by the user, so we call this a secret signature, which provides non-repudiation and integrity check, and also message confidentiality. For example, the server can sign a session key K_{AB} by using a private key matching this secret public key, assuming A knows the value of ST_A.

It seems to be secure against off-line password guessing attacks, and it also provides forward secrecy, meaning that the exposure of a past session key will not reveal any information about the user password.

Ford Long Wong: Does the use of the password actually give you much, because presumably the entropy of the password is pretty small, so the public key can be found.

Reply: It depends how you perform your off-line attack, I guess, even though the entropy of the password is small. But if there is nothing that will verify whether your guessed password is correct or not, then it wouldn't leak any information to the attacker.

Bruce Christianson: If the argument is that the attack is through the normal rules where the attacker only gets one false guess, then he can't attack on-line.

Reply: Yes, definitely that is something.

Bruce Christianson: Can you remind us what property this modified protocol has that Gong's optimised protocol doesn't?

Reply: They are using standard block cipher.

Ross Anderson: Or you can use a package transform[1], or something more modern.

Reply: Here we use asymmetric encryption rather than a standard block cipher.

Ross Anderson: Well I suppose the obvious criticism is that people don't use EKE because it's patented. People might have difficulty using this because of the Boneh-Franklin patent, so can you produce a version of it that only uses an identity-based signature, which would then be public domain?

[1] Rivest: LNCS1269, pp210-218 All-or-nothing encryption and the package transform; see also Anderson, LNCS3957 pp 231-245.

Bruce Christianson: Yes, it's the order in which the claims fail, but that's an awful lot of waiting.

Srijith Nair: The initial part of the work when you created a public key, you take the password, and the hash, and all that, I'm not sure about how much of this has already been proposed in other papers. There is something called anonymous identity-based key issuing, or something like that, but they haven't taken it to this level, they have used this for key issuing in anonymous ways. They chain two keys and use one, and then activate the other, but the way they create the public key is using this hash function.

Kenny Paterson: Well every identity-based system uses a hash function to create a public key to let someone into the system.

Reply: Yes, that would be always the case.

Bruce Christianson: But hashing together like that to produce not just public keys but capabilities, is already a move on to something new that's more like what we want.

Chris Mitchell: If I understood this properly, it is a password-based authentication scheme, and there are a lot of other schemes that have been published since the original paper in the early 90s, and indeed we have an ISO standard, which is now finished, we've an IEEE standard, which is nearing completion, with about 20 mechanisms in it, why should we use yours as opposed to all the others?

Kenny Paterson: This is an interesting piece of research; if we only worried about ISO standards when we were doing research, we probably wouldn't bother doing anything better.

Bruce Christianson: But what you're coming up with here is a recipe that lets you get a vast number of different schemes just by plugging different algorithms in, and which gives you the knowledge that there's no protocol type weakness that somebody's able to exploit, that's the point.

Reply: Yes, exactly.

Jun Li: I remember you said that one of the properties of Li Gong's paper is the secret public key works in the situation where the user doesn't know the authentication server's public key. But in the ID-based scheme, with revocation, the user has to know some system parameters which actually include the server's public key.

Reply: Depends how you look at it, but that's a very good point. I would say that is one of the limitations of this approach because you need to distribute that public key.
Jun Li: Could I say, in that case, if you have to authenticate the server's public key, your public key actually is not a secret public key anymore.

Reply: I think that's different, it's the system parameters where the users need to authenticate beforehand, but then after that, all these secret public keys are produced fresh, and they are different.

Jun Li: So actually your fresh key doesn't satisfy Li Gong's requirement, in terms of the situation, the computer doesn't know the server's public key?

Reply: Yes, if you look at it that way.

Bruce Christianson: But to be fair, almost none of the other EKE schemes on the market, consider revocation in any serious way at all: it's not fair to single him out.

Reply: Thanks very much. [Laughter]

Vintage Bit Cryptography

Bruce Christianson and Alex Shafarenko

University of Hertfordshire

We propose to use a Random High-Rate Binary (RHRB) stream for the purpose of key distribution. The idea is as follows. Assume availability of a high-rate (terabits per second) broadcaster sending random content. Members of the key group (*e.g.* {Alice, Bob}) share a weak secret (at least 60 bits) and use it to make a selection of bits from the RHRB stream at an extremely low rate (1 bit out of 10^{16} to 10^{18}). By the time that a strong key of reasonable size has been collected (1,000 bits), an enormous amount of data has been broadcast (10^{19}–10^{21} bits). This is 10^6 to 10^8 times current hard drive capacity, which makes it infeasible for the interceptor (Eve) to store the stream for subsequent cryptanalysis, which is what the interceptor would have to do in the absence of the shared secret. Alternatively Eve could record the selection of bits that correspond to every value of the weak shared secret, which under the above assumptions requires the same or greater amount of storage *i.e.* $2^{60} \times 10^3$. The members of the key group have no need to capture the whole stream, but store only the tiny part of it that is the key. Effectively this allows a pseudo-random sequence generated from a weak key to be leveraged up into a strong genuinely random key.

The stream observation time given a 10Tbit/sec broadcast rate is only 10^6 to 10^8 seconds, or a week to a few months. Over this time the shared secret is not used for any kind of communication and so the only possible threat is insufficient key storage security, which is present in any cryptographic scheme. It is interesting that in our approach the passage of time strengthens the resulting key: the longer we wait before the key is used, the less chance there is that any relevant part of the stream is present in a storage facility anywhere in the world, due to the sheer mass of data. This is, in a way, opposite to the standard assumption of cryptographic strength, that keys becomes weaker with time. Accordingly, we call this system Vintage Bit Cryptography.

It is interesting to note that vintage bits are not a hostage to future technology development: the ability to record more data per unit cost in future has no influence over the present time: vintage bits not recorded now will not become available later. Nor does leaking the weak secret compromise vintage bits obtained earlier, provided the time difference is sufficient to overwhelm the capacity of attacker's stream storage. In particular, schemes such as EKE [2,4] can be used to leverage the initial weak secret into a strong pseudo-random seed without fear that subsequent development of quantum computers (allowing the easy solution of discrete logarithm puzzles) will expose previously obtained vintage bit keys.

Beacon systems have been proposed before [9,12,10], particularly in connection with satellites [13]. A traditional beacon implementation based upon a geostationary satellite would make the key distribution system available over a wide area at a very small cost to a consumer. But at present digital broadcast satellites lag far behind optical fibre in terms of bandwidth, transmitting only on the order of 10Gbits/sec, although this rate will increase with the use of higher microwave bands.

A satellite solution which could prove more interesting is a swarm of micro-satellites in a Low-Earth-Orbit (LEO). Such satellites could be equipped with an array of tuned silicon lasers that transmit on a number of wavelengths, and with physical random bit generators that control the lasers. Importantly, no radiation protection is required in this case. Indeed, the spacecraft need not have any processing power since all it broadcasts is random digital noise. LEO satellites could be tiny: less than a cubic decimeter undeployed size, with a small production and deployment cost: space scree (rather than dust).

Anyone with a few tens of thousands of dollars to spare can already have micro-satellites launched using a non-governmental space operator. These satellites can keep orbit for years without thrusters and can maintain their orientation by purely passive means. The overhead passage for one of these craft would last 20-30 min, so a continuous RHRB stream at terabit rates would require a hundred spacecraft or so. Using a polar orbit one can ensure that the continuous stream is available anywhere on the planet, and that the area of consistent observation (where all ground observers can see the same satellites at the same time), is of the order of 1000km across, which makes it quite suitable for European applications in particular. The XORing of streams produced from several satellites launched by mutually distrusting parties eliminates the need to trust any individual craft.

However optical fibres are an attractive alternative to satellites, and our primary interest in this paper is with very high bandwidth fibre-optic beacon systems. The first implementation issue to consider is feasibility.

A single optical fibre can already carry more than 1Tbit/sec with a bit-error rate (BER) better than 10^{-3} using an appropriate combination of Wavelength Division Multiplexing and Optical Time Division Multiplexing. Low BER is a key goal of conventional fibre optic communications, but this very tough restriction is not an issue for us. Transmission errors are easily mitigated against by using a simple protocol based on FEC and cryptographic hash functions:

$$A \longrightarrow B : P|Q$$

where $K = K_1|K_2|K_3$ are the vintage bits recorded by A: K_1 is the eventual shared secret with B, K_2 and K_3 are used as one-time pads;
h is a strong hash function, $P = K_2 \oplus h(K_1)$;
and F is a forward error correction function, $Q = K_3 \oplus F(K_1|K_2)$.

The protocol succeeds if B's calculated value for $h(K_1)$ based on the value of $K_1|K_2$ recovered from Q agrees exactly with the value for $h(K_1)$ recovered from P. Note that the message $P|Q$ can be sent over any open, moderately non-lossy channel: no endpoint authentication is required, and data integrity is an issue only if we are concerned with denial of service attacks. In particular, if the message is broadcast, the identity of Bob need not be revealed.

Because low BER is not a consideration for vintage bit cryptography, we are able to propose the use of cheap optical fibre technology which is not suited to the mainstream communication industry. This provides an attractive (cheap!) alternative to the optical fibre systems already being used for key distribution in industry, which use very low bit rates and quantum technology. These quantum-based systems make eavesdropping detectable, but come at a very high cost [6,7,8,14,15,16]. This form of quantum technology also depends crucially on the physical integrity of the optical cable: it eliminates passive eavesdropping but avoiding the man-in-the-middle attack requires at least a weak form of end-to-end authentication for the side-channel, which imposes constraints similar to the initial sharing of a weak secret in our proposal.

Ensuring the integrity of the communication path from a shared beacon is problematic with fibre-optic technology (in contrast with satellites). One simple possibility in the case of a point-to-point link is to co-locate the beacon with one of the participants (say Alice), as may be done in the quantum key agreement scenario. However a more interesting case is where we wish a single beacon on a fibre optic loop to be shared by all the loop nodes. In this case we would like to reduce the integrity requirement to reliance merely upon the integrity of the beacon itself, and not that of the fibre optic medium.

One possibility in this case is for clients to pre-share a weak secret with the beacon (or more accurately with a co-located trusted server). As they collect vintage bits to share with each other, Alice or Bob uses this weak secret to generate bits shared with the beacon service, over the same observation period and using the same protocol. The protocol between Alice and Bob now succeeds only if the vintage bits shared with the beacon have not been tampered with: if they are correct then the real beacon is the source of the bits shared between Alice and Bob. Otherwise the bits are corked and should not be used. The bits shared with the beacon can be discarded, or used to update the weak secret shared with the beacon. Optionally, the beacon service, since it is trusted anyway, can be used to share an initial secret between Alice and Bob in case they have not already been introduced.

However it may be a disadvantage for a beacon protocol to require per-client state to be kept at the server end, and individual communication between each node and the server along the side channel. An alternative is to use a variation of a Merkle-type protocol [3], combined with an additional lower-bandwidth authenticated broadcast by the server. In this case, whenever Alice and Bob collect vintage bits, at least one of them also takes a larger random sample of the beacon, at a rate of order 1 in 10^8–10^9. The beacon server also certifies (for example by public key signature or hash pre-image [1,11]) a random sample of the broadcast taken at a similar rate, which it publishes following sufficient delay to guard against the possibility of a replay attack. The beacon can sample blocks randomly, rather than individual bits. Alice or Bob can now guard against a false beacon by verifying (say, more than 80% match) sufficiently many of the bits which by chance occur in both server and client samples over the course of the collection period.

The number of shared bits increases linearly with the size of the sample being collected. Sampling at a rate of 1 in 10^8 for a base transmission rate of 10Tbps will thus require the beacon to certify about 1Gbyte per day. (If Alice also samples at the rate of 1 in 10^8 then over 80 Merkle bits will be shared per day.) There would be no technical difficulty for the beacon to send this amount of data down the optical medium given the terabit rate of the system. The beacon sample should be broadcast along with a sufficiently long hash, which is signed for authentication. However there is no real-time restriction on the broadcast of the signed hash, which may take place offline. The clients need to know that the beacon was authentic only before they commit to using the newly collected shared key, which as we indicated above takes a few weeks to a few months. This time scale also makes it feasible to employ authentication based on physical security (*e.g.* the delivery of physically authenticated records on tamper-evident media to the clients' sites) as an alternative.

The trust assumptions in our fibre-optic approach are very limited, and are nearly the same as those of the competing quantum approach: the beacon has to be trusted to be authentically random, and a man-in-the middle attack must be detected by end-to-end use of a weak secret. However we make no assumptions about the physical integrity of the fibre-optic link.

While the idea of cryptographic use of a beacon is not in itself new, previous work has tended to focus upon satellite implementations. The threat model for the fibre optic context introduced here is rather different to that for the satellite, and the ramifications of this should lead to interesting new developments.

References

1. Anderson, R., Bergadano, F., Crispo, B., Lee, J.-H., Manifavas, C., Needham, R.: A new family of authentication protocols. Operating Systems Review 32(4), 9–20 (1998)
2. Bellovin, S.M., Merritt, M.: Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In: Proceedings of the I.E.E.E. Symposium on Research in Security and Privacy, Oakland (May 1992)
3. Christianson, B., Wheeler, D.: Merkle Puzzles Revisited – Finding Matching Elements between Lists. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2001. LNCS, vol. 2467, pp. 87–90. Springer, Heidelberg (2002)
4. Christianson, B., Roe, M., Wheeler, D.: Secure Sessions from Weak Secrets. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2003. LNCS, vol. 3364, pp. 190–205. Springer, Heidelberg (2005)
5. Ding, X., Mazzocchi, D., Tsudik, G.: Experimenting with Server-Aided Signatures. In: Proceedings of Network and Distributed System Security Symposium, NDSS 2002 (2002)
6. Gisin, N., Ribordy, G., Tittel, W., Zbinden, H.: Quantum cryptography. Rev. Mod. Phys. 74, 145–195 (2002)
7. Gobby, S.C., Yuan, Z.L., Shields, A.J.: Quantum key distribution over 122km standard telecom fiber. Appl. Phys. Lett. 84, 3762–3764 (2004)
8. Hughes, R.J., Morgan, G.L., Peterson, C.G.: Quantum key distribution over a 48 km optical fibre network. J. Mod. Phys. 47, 533–547 (2000)

9. Maurer, U.: Conditionally-perfect secrecy and a provably-secure randomized cipher. Journal of Cryptology 5, 53–66 (1992)
10. Cachin, C., Maurer, U.M.: Unconditional Secrecy against Memory-Bounded Adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 292–306. Springer, Heidelberg (1997)
11. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988)
12. Mitchell, C.J.: A storage complexity based analogue of Maurer key establishment using public channels. In: Boyd, C. (ed.) Cryptography and Coding 1995. LNCS, vol. 1025, pp. 84–93. Springer, Heidelberg (1995)
13. Rabin, M., Ding, Y.Z.: Hyper-Encryption and Everlasting Security. In: Alt, H., Ferreira, A. (eds.) STACS 2002. LNCS, vol. 2285, p. 1. Springer, Heidelberg (2002)
14. Wu, B.B., Narimanov, E.E.: A method for secure communications over a public fiber-optical network. Opt. Express 14, 3738–3751 (2006)
15. Yoshizawa, A., Kaji, R., Tsuchida, H.: 10.5 km fiber-optic quantum key distribution at 1550 nm with a key rate of 45 kHz. Japanese J. Appl. Phys. 43, L735–L737 (2004)
16. Yuan, Z., Shields, A.: Continuous operation of a one-way quantum key distribution system over installed telecom fibre. Opt. Express 13, 660–665 (2005)

Vintage Bit Cryptography
(Transcript of Discussion)

Alex Shafarenko

University of Hertfordshire

This may be a highly controversial talk, because this is an area where things are periodically rediscovered. But in the process of reinventing it, I think we've found a few interesting protocol issues, and a few interesting technological issues, which make it worth revisiting.

What's the idea of vintage bit cryptography? It's not an established term, it has other names, but the key idea is that a quantity of information too large to be stored anywhere on this planet is effectively an unusable secret. What's a secret? A secret is something that nobody has a copy of, or no bad guy has a copy of, right. Certainly nobody has a copy of this secret because it's too large to be copied, and it's unusable for the same reason. So the key principle is: public transmission of the unusable secret, with subsequent private selection of a small usable sub-secret. So secrecy comes from two things, the fact that you can't store the whole thing, and that you don't know which random selection of it has been taken.

James Heather: How can you transmit the thing if it's too big even to store it?

Reply: By generating it randomly. It's too big to store, but it's not too big to transmit over a significantly long period of time. The amount of information too large to be stored is technology dependent; this whole approach is technologically dependent. In a sci-fi world, if you have 10^{20} bits of storage on a PC, this approach doesn't work, at least at the transmission speeds that we have available at the moment. But the interesting thing is that the paper by Mitchell[1] goes back to 1995, and he gives some figures about the cost of storage, transmission rates, etc, and the figures that we have today still support the principle. Ten years in this area is a huge amount of time.

So here are some assumptions. I think the assumptions of this method are more important even than the method, because there are several variations. The first assumption is the availability of an open authenticated channel between all members of the key group. We're talking about key distribution: this protocol is used for agreeing a secret key within a group of people. So they need to have an authenticated channel, which doesn't have to be secret, confidential, it could be an open channel. The second assumption is that the public broadcaster has high entropy, so is genuinely random. If it doesn't have high entropy then it can be compromised easily by knowing the information basis of the broadcast from

[1] Referenced in the position paper.

which the whole broadcast can be reconstructed. If you can compromise the authenticated channel, then the original approach collapses, so in that approach you need to secure the authenticated channel very well. (We'll show later how to remove that assumption.)

So how is it done? The original method was to publicly agree a long observation period, then Alice and Bob collect a random selection of m bits each, and they don't tell anybody which bits these are as they collect them. Then they use the open authenticated channel to check which bits are common between them. Now if you do the sums then you will see that quite a lot of bits that they collect are common between them; if they collect m bits each out of the M that are broadcast then the number of bits that they have in common is about m^2/M. These bits form the secret key. They openly tell each other which positions they have in common because the large unusable secret is already gone, it can't be stored, and if you didn't know which bits to collect you wouldn't have collected them.

If there's an interceptor, Eve, then she also has her own random selection, say of n bits. These bits would have bits in common with both Alice and Bob, but the proportion of those would be small, assuming that M is such a huge number that you can't collect that many bits, so n would be much smaller than M. So Eve doesn't get very much; for practical purposes Eve gets nothing at all with a high probability. This is wonderful because no secret communication between Alice and Bob takes place, and yet they've agreed a common secret key, and nobody can intercept that, and the only thing that we need to secure is an open authenticated channel.

The first thing that comes to mind is that this is not a very scalable procedure because if you have lots of pairs of users exploiting the same public broadcast — and the public broadcast is an expensive thing, so you would want to share that — then what's the probability that an arbitrary user can intercept the private key of a pair that he doesn't belong to? We all know about the birthday attack, so this is not a small number necessarily, and the public source is expensive so we would like lots of people to share that.

Now with protocol issues, the original paper admits that if the open authenticated channel is there then you can use Diffie-Hellman and agree a private key, so what's the point of having a public broadcast? There is a point, still, because Diffie-Hellman may in the future be broken (quantum cryptography, all the rest of it), and the proposed scheme is free of all that, it gives you genuinely random keys. However, if you do have an open authenticated channel of good quality, you can assume at least that Diffie-Hellman will not be broken over the observation period, which in the original paper is one day. Also it's one day so you're not short of time; you can spend an hour of that day to run a really huge Diffie-Hellman, which is hard to break even if you have some trick up your sleeve. And the complexity of Diffie-Hellman is linear in the size of the key. So, maybe all you need to do is to have a common secret, a small shared secret between Alice and Bob, and use that for a common random selection, then you don't need to rely upon the collisions between selected bits, and so that makes it unnecessary to use any protocol subsequently, you just share a secret, and after an observation period you have a strong key which is common to Alice and Bob.

Another reason to make this assumption is that an open authenticated channel (in the absence of the trusted third party, public key infrastructure, and all the rest of it), does involve sharing some sort of secret, so a shared secret is already assumed in a way. If you don't want to use public key cryptography, why not use the shared secret for a common random selection.

But there's a much greater problem. In fact it is, strictly speaking, insuperable in the scheme that's being proposed. It is impossible to prove that the public broadcast has sufficient entropy. A bad guy can replace the public source by a random number generator, note the seed of the random generator, wait for the shared secret to be used to collect a strong key, and then use the random generator again to recover the strong key completely. And it is impossible for the broadcaster to prove that it is genuine, that it doesn't use any random, pseudo random generator, and it is possible to disprove by the user. So you need a technological solution. I can't see any kind of cryptographic solution here.

Another problem is the fact that high bit rate sources (and we need one for public broadcast here), are prone to errors. So Alice and Bob will not have exactly the same bits from the same random selection; they will have a large key with a high proportion of bits common, but some not: maybe up to 10% errors, if we use the technology that I will touch upon later. So you need some sort of protocol to agree a common key when you have *almost* coincident bit strings.

Well in fact it's not difficult. Here's your bit key $K = K1|K2|K3$ that you've collected from vintage bits. The part that you want to be the shared secret key is $K1$, then you sacrifice $K2$ and $K3$. Alice calculates a forward error correction check-sum $F(K1|K2)$ based on her value of $K1|K2$. That FEC check-sum would be such that it could correct a large number of errors — it has enough redundancy. We don't want to disclose any information about the secret key so Alice XORs the FEC with her value of $K3$. This may not be the same $K3$ as for Bob, because $K3$ also has errors. But fortunately the forward correcting algorithms do not require you to have a clean check-sum; a check-sum can also have errors, and that doesn't prevent their recovery. All you need to be worried about is that the strength of the error correcting algorithm is sufficient to recover from errors in both.

OK, so $\longrightarrow B : FEC(K1_a|K2_a) \oplus K3_a$. If the strength of error correction is sufficient then B can compute A's value for K1, and this is the shared key. Now we need to verify that it's the same key, that the error-correction has worked. We can do this at the same time as removing the requirement for an authenticated channel by including a strong cryptographic check-sum, so $A \longrightarrow B : h(K1_a) \oplus K2_a$. If the first step worked then Bob has the same value as Alice for K2, so he can recover Alice's value for $h(K1_a)$ and compare this with the hash of his own recovered value for $K1_a$. So at the end of this the protocol either fails, because there were too many errors, or it succeeds. Now the technology that we'll talk about a bit later is reliable enough that you can almost guarantee that it always succeeds.

OK, so what's the technology? I collaborate with one of the leading fibre-optic groups in the world, I think. At Aston University they've got a huge experimental set-up worth millions, and they also do lots of theoretical research, and so I called them up three days ago and said that I needed something fast, and not necessarily reliable. They said, well that's a problem, we can give you 5 terabytes per second, but no better than 10^{-3} bit error rate. I said, I'm happy with 10^{-1} bit error rate. They said, well that may be 10 terabytes per second, but we won't go further than across the Atlantic. I said, I don't mean across the Atlantic, I mean locally, between the main headquarters of the bank, and the branch, maybe 200 kilometres. They said, well we haven't researched that, maybe 100 terabytes per second. They just split the optical range in the fibre in 2000 channels, and they send 5 gigabytes per second down each channel, and they're very worried about interference between different channels. If I want a random stream then I needn't be worried about that, because it can't get less random due to that.

Audience: I just want to query the idea that it can't get less random. If you have the sort of interference where one channel is completely wiping out another channel, then that would make it less random.

Reply: Ah, that's not what happens. What happens is that the bit error rate, which is normally 10^{-6}, becomes 10^{-2}. It's a non-linear medium, highly unpredictable, and there's also noise from amplifiers, from repeaters, from all sorts of things; there's a huge amount of technology sitting behind it.

So that's very interesting for them, nobody asked them before for a bad fibre which is fast, people usually ask them for a good fibre, and don't mind it being a bit slow.

The other problem that they find with optical fibres is that they're bursty, errors come in clusters. We absolutely don't care about that either, because we make a very rarefied selection, we take one bit out of 10^{17}.

So what's feasible storage? People from the Grid project can correct me, but last time I checked... there's this European collider project which requires the computational grid to store collision data from a year's worth of observation, and they reckon to need 10 Petabytes, and that's huge. OK, let's assume 100 Petabytes is unfeasible, just for the sake of the argument. Then the question is, how weak is the weak secret? If the weak secret is very weak, then I can just do the observation for each value of the secret, instead of observing the whole stream. Basically if you have N bits in the broadcast, and you need k bits of key, then assuming that you have a good random generator, you need $\log_2 N - \log_2 k$ bits of weak secret. Even under aggressive assumptions 60 bits would secure vintage-bit cryptography. What's 60 bits — a 10 character password. They can share a 10 character password, then after the observation period they can publish that password, right, because you can't go back in time and collect those bits that you needed to have collected.

George Danezis: So how could you actually generate good randomness of the public broadcast?

Reply: Oh, well, just use a resistor, heat it up, and trigger some digital device, you will get good randomness; I promise you can't repeat it.

Michael Roe: I know that type of device, and it turns out to be quite hard to get an acceptable number of random bits out of them because it amplifies all kinds of horrible things and you end up having a sort of driven oscillator.

Ross Anderson: That's a separate engineering problem about which much is known, but the way I see this as you present it, you're not actually doing broadcast randomness, you're really competing with the quantum crypto guys, who say, give us a fibre from London to Geneva, and we'll send you a key. That is what makes this different from Maurer.

Reply: Yes, yes, that's exactly right. I have a slide about that at the end, because there's more than one very interesting technology that fits the vintage bit sort of scenario.

We need a countermeasure for the low entropy problem. What can we do? We can use a reflector instead of a public source, so each guy in the key group generates their own randomness at a smaller rate, if you have a thousand users, then you just generate one thousandth of the public broadcast bandwidth.

Mike Bond: What do you mean by reflectors? Are they an abstraction?

Reply: No, it's a physical device. The reflector just combines streams from these sources.

Mike Bond: So it's a mirror?

Reply: No, it's a transponder essentially. It takes information from all these channels, which are different wavelengths on the same fibre, for instance, and then interleaves them. If it doesn't have enough sources it can interleave some random sources as well, of it's own doing. The more users you have for this system, the better it works. That's one half of it. The other half is a monitoring loop, each of the users computes mutual information between the bits that were sent to the reflector, and the bits that were received in the same positions. Suppose there's a random interleave, for instance, using a public formula, so we know where our bits are. The reflector can't guarantee that these bits will be intact, because there are users that will collide with those bits, or there may be also some random content that will collide with this, but we can measure the amount of mutual information, and if it drops below the critical level we can raise the red flag, after the observation period. The crucial thing here for safety of the cryptographic solution is that we don't communicate at all during the observation period, because we want to make full use of the huge shared secret. OK, now just note that neither the reflector, nor the channels to the reflector, are trusted. This doesn't matter; you can intercept everything, you can forge anything you want. Because of the monitoring of mutual information, you will be found out eventually. It's statistical of course, it's all probabilistic.

Now quantum key distribution has been mentioned, and Toshiba is selling a solution; it's no longer in the lab, it's a product. You have a piece of fibre, and a technology that guarantees that a single photon emitted by the source is received by the receiver without anybody intercepting it. If somebody intercepts it, this will be detected. The cost of this technology is tens of thousands of euros, depending on what you buy, per user. It has a reasonably high performance — it actually transmits about 100 secure bits per second, which for cryptographic applications is quite a large number. Bruce drew my attention to the fact that it is actually quite prone to the man-in-the-middle attack, just cut the fibre, get the man-in-the-middle, you're done. That's why they need integrity and end-point authentication on the side-channel. You could continuously monitor the fibre, using all the other measures that you can take to prevent man-in-the-middle, but if you can do that then you don't need quantum key agreement in the first place. In fact we all know a power failure goes a long way in these schemes!

There's also talk about satellites. Why can't we have a satellite broadcaster up in the sky and use it as a source of vintage bits? In fact we can't. My first instinct was to use a TV broadcast satellite, it's got about a thousand TV channels, each channel around 2 Megabytes per second, so this is in the order of 10 Gigabytes per second. The problem is that all the contents of the broadcast is recorded somewhere, and if you need a selection you just go back in time and ask the content providers to give you a copy. However, there's new and exciting technology called UWB, you probably have heard about it, because it's going to be used for PCs, short-range communication, but it is particularly good for satellites.

We are coming to the period in technological development where the term "frequency" will fall out of use. Now the content will carry itself, there's no carrier. What will happen is, you will have pulses of electromagnetic energy, 200 pico seconds in length (that's a very short pulse), with a duty cycle about 64K, so there will be 64K slots, on average, between two pulses. The amount of energy that accumulates in one pulse is such that if you just consider these pulses, you could hear them at the end of the solar system. But, because you hear noise in between, you can't actually hear the pulses unless you know exactly where they are. So the same principle applies, the pulses are positioned according to some sort of random sequence, if you know that sequence you can hear. By the time you've cracked the password you've already missed all the bits, so it works on the same principle.

Now if you don't want a geostationary satellite being used as a source for reasons of entropy (because it could be compromised, an evil government can control it), then to get free of this problem you can have a swarm of micro-satellites on Low-Earth-Orbit. You have a satellite orbiting the earth not very far up, like 100 kilometres, and now the earth will rotate underneath. Now if you have, say, 40 satellites here, in the same orbit, then what happens is that you will always have a satellite overhead. The footprint of this scheme is about three or four hundred kilometres and within that area all the users will see the same satellites. Now each micro-satellite can be completely dumb, it could be mass produced, it doesn't even have to have any kind of intelligent electronics, or radiation protection, because all it needs to do is create digital noise —

not analogue noise because that's hard to deal with, but digital noise. Now to compromise this scheme, you would have to compromise a significant number of satellites because users can XOR several of them. You can have not 40, but 400, in that orbit. They are the size of a grapefruit and the cost of launch is about 10,000 euros per kilogram, which is competing with the quantum distribution fibre-optic solution from Toshiba. The satellite itself will be free, though, because it has no intelligent satellite guts in it. It's just a simple electronic circuit, which actually is faulty as well, because it's not radiation protected; but that's OK, you can't get more random than random. And I don't think this is compromisable by any kind of realistic means. In science fiction you just fly your spaceship to each of the satellites and replace it, but since NASA tracks all satellites the size of grapefruit and above, this will be known; if you just touch it, it changes orbit.

OK, conclusions. Over the last ten years, the storage to speed ratio has not changed much. We can still do vintage bit cryptography, and that encourages me to suggest that maybe we will be able to do it for the foreseeable future. Despite the fact that I've shown two schemes, I don't think satellites are a good solution because of the cost of management implications, and trust issues. However, I must say that a fibre-optic broadcaster is entirely possible, to the extent that we're going to construct one, with the guys from Aston University, and demonstrate it.

Bruce Christianson: We think we can undercut the quantum fibre optic product.

Ross Anderson: You don't need a broadcaster, you need a fibre-optic link?

Reply: Yes, a broadcaster in fibre-optic, so there will be one source and lots of receivers along the same fibre.

Mike Bond: Sorry, did you assert earlier on that your scheme is invulnerable to man-in-the-middle in the same way as the Toshiba scheme is, or that it isn't?

Reply: I did assert it given that we have a reflector on the source[2]. The first experiments will be with the source, then we'll try and engineer a reflector.

Mike Bond: Using cable cutting between people and the reflector, why can't the attacker assemble everybody into a virtual subnet with 49 other imaginary people, and 50 reflectors? So everybody talks to their own reflector, and 49 other fake people?

Reply: The amount of mutual information that the key group receives from the reflector is known by calculation. We monitor what the reflector throws at us, and any random sample from any other people...

Mike Bond: But each person's monitoring their own?

[2] In fact, even without any refectors it suffices to have integrity and source-authentication on a low-bandwidth broadcast side-channel from the random source. This is explained in the position paper.

Bruce Christianson: I understand what you're saying: the channel over which the protocol runs — you remember the XOR and the forward error-correcting scheme — that's not over fibre-optics, that's end-to-end coverage.

Ross Anderson: You need some authentication somewhere.

Bruce Christianson: Yes, we do, but so do the quantum people. You need to know that you're listening to the correct source. This typically involves a conventional side-channel.

Ross Anderson: Which brings us back to the problems of having the authentication end-to-end. We know the quantum crypto guys have different ways of doing this, by looking at hashes of sub-streams they receive and checking that the hashes are the same; presumably at least one type of bootstrap from password will do that. You can't do it many times from the same password though.

Bruce Christianson: The key point is that the quantum people have this same problem, and there we can use the same techniques that they're using. They have to have a weak shared secret for authentication. For the next authentication, we can use some of the new strong secret.

Reply: Actually, for our scheme the communication between Alice and Bob doesn't need to be authenticated at all, it only needs to be authenticated if you want to deal with denial of service, for no other reasons.

Bruce Christianson: Yes, that's true.

Michael Roe: Don't you also have to know that the secret that you end up with is shared with the person you think it's shared with, rather than with the attacker?

Ross Anderson: The authentication there is implicit, because the authentication is not going to work if you chose different sub-streams of the random source. What you're demonstrating here then is yet another way in which to parlay a weak shared secret into strong authentications, namely using your mechanism of very long bit streams, and transmitting sub-streams with their error correction bits.

Bruce Christianson: Unlike the quantum people, we get that for nothing.

Ross Anderson: OK, so you should bring out in the paper that you have got yet another alternative to EKE.

Bruce Christianson: Yes, that is a good point. I think the other point worth making is this: the first step is for users to get from the weak secret to a strong secret, and then to use the strong secret where they would have otherwise used the weak one to authenticate. That way they can do it as many times as they like.

Ross Anderson: Yes, OK. So perhaps what one ought to do is write this out formally as a paper, and point out that you've got an error correction-based protocol for authentication.

Bruce Christianson: Yes, that's a good way of putting it actually, because that makes the novelty clear.

Audience: A second question I had was about your storage requirement assumptions. When you said 100 petabytes, I remember doing sums for a look-up table for DES, which was about 500 petabytes for a single ciphertext. I was thinking, gosh, I wonder if the NSA has got 500 petabytes. My question is, given the special requirements of keeping the data just long enough to be able to look back and get the bits you want just in time, could there be any specialised storage, for instance, like the equivalent of mercury delay lines set at solar system scale, or could you send it all into space?

Bruce Christianson: Or use slow glass[3].

Mike Bond: Just keep it spinning in optic-fibres for long enough.

Bruce Christianson: Or bounce it off a deep space probe, and send it back again.

Reply: The observation period is not necessarily limited to one day; the longer it is, the more insuperable the acquisition problem becomes.

Mike Bond: What's the regional range for satellites at the moment, presumably only a few light hours?

Bruce Christianson: There's some probes that have already left the solar system. And maybe aliens will reflect our broadcasts back at us[4].

Reply: You won't get my signal back. This is my first attendance at a security protocols workshop, and Bruce warned me to expect intellectual paranoia. But this is paranoia on a galactic scale. You need a 70 metre deep-space network dish to communicate with something that's that far.

Bruce Christianson: The key point is that you need to have already launched the probe some time ago.

Reply: Yes, the whole strength of this approach is that it is retrospective, all of it, yes.

Bruce Christianson: The advantage of this approach is that magic buttons invented tomorrow don't help the attacker against bits you have already laid down.

Reply: Exactly, you need a time-machine to break this; we could call it time-machine security.

[3] Bob Shaw, Light of Other Days, Analog, August 1966. In the story the refractive index of slow glass is about 1.5×10^{19}, as light takes 10 years to travel a quarter of an inch. Lene Vestergaard Hau *et al.*, Nature 397(1999) p 594 describe real slow glass with an RI of 3×10^{10}, about 120 feet per hour. However for real slow glass the product of delay with bandwith is fixed for a given cross section, so it is still worth investing in a picture window, rather than a single thin fibre.

[4] Probably starting with the BBC Third Programme.

Jolyon Clulow: I'm still not clear what the difference is between this and Michael Rabin's proposals for a fleet of satellites.

Bruce Christianson: Well, we didn't say much about the fibre-optic case in the talk[5], the short answer is that the threat model is different for fibre optic. But in all cases the trick that makes it work is that you can't store all the potential key material at once.

Ross Anderson: So what precisely are the security semantics of strings that are too long to store? We've seen now several examples of things that we can do with them. Suppose you have got an Oracle, which is privileged over normal mortals in that it has infinite memory, what special tricks can we make this Oracle do, what sort of new complexity tasks can you conjure up to keep the theoreticians busy for the next 30 years?

Reply: What a wonderful thought, that's a good question to end on.

[5] Largely because we hadn't yet filed the patent application.

Usability of Security Management: Defining the Permissions of Guests

Matthew Johnson and Frank Stajano

University of Cambridge

Abstract. Within the scenario of a Smart Home, we discuss the issues involved in allowing limited interaction with the environment for unidentified principals, or guests. The challenges include identifying and authenticating guests on one hand and delegating authorization to them on the other. While the technical mechanisms for doing so in generic distributed systems have been around for decades, existing solutions are in general not applicable to the smart home because they are too complex to manage. We focus on providing both security and usability; we therefore seek simple and easy to understand approaches that can be used by a normal computer-illiterate home owner, not just by a trained system administrator. This position paper describes ongoing research and does not claim to have all the answers.

1 Introduction

1.1 The Smart Home

Of the many possible applications scenarios of ubiquitous computing, numerous visions of "Smart Home" environments have been put forward [1,2,3,4,5]. In a smart home, everyday objects such as appliances and furniture, as well as systems such as heating and ventilation, will feature embedded processors and communications and will work as both sensors and actuators in an integrated home system. On the communications side, low power and low bit rate wireless networks suitable for control of home systems are the focus of an industrial consortium[1] and an IEEE standard [6]. These models address multiple embedded devices and control of them from authenticated principals. Many of these models also propose solutions for preventing usage by strangers. What have not been adequately addressed so far are the issues of guests.

Unlike the people who normally live in the house, guests should not have access to everything; and, to the extent that regular users are modelled as having "accounts" on the home system, guests should not have to be given accounts; however, they are not strangers either. There are many common situations where it is desirable to give some level of access to a guest. For example, a guest might wish to play some music from their portable device on your Hi-Fi or a movie on your TV, and you might be happy to allow this; given the technical capabilities

[1] Zigbee: `http://www.zigbee.org/`

of a smart home with a wireless network, it would be unreasonable if this were not possible. But you shouldn't for that be forced to register the guest as if they were a new permanent occupant.

You would not want guests to be able to change your play-lists, central heating timers, intruder alarm codes and so on, although you might want them to be able to upload a song for you to listen to later.

An example of a commonly used delegation procedure in pre-smart-home situations is to leave the guest with a key until they leave and have them post it back through the door. When taking this as a model, you might want to be able to guarantee that the guest's access rights expire at the correct time and cannot be copied and used later.

One of the biggest challenges is not just creating a system that can do all this, but creating one which is easy to use. Systems which are designed to mimic actions in the real world (such as lending a key, or letting someone play a CD) need to match people's expectations and conceptions for them to be convenient and easy to use.

1.2 Location Awareness

In the majority of context-aware systems, the primary source of contextual information is the relative (and sometimes also absolute) location of the people and objects involved in the interaction.

There are a great variety of ways of obtaining location information; Hightower and Borriello [7] as well as Beresford [8] provide useful surveys and taxonomies in the field.

One possible distinction is between systems, such as the Active Bat [9], that provide absolute positioning of objects within some local reference frame, and systems where active objects can just sense, perhaps through radio signal strength, whether they are within a given range of each other. In the first case, a central facility can trigger events based on the positions of the objects. In the second case, instead, objects have only a vague idea of their position and so even guaranteeing containment of a device within a room is difficult.

A smart home setup that will work in the latter case will also work in the former; we will concentrate on location systems without a strong notion of position so that our model will support both.

1.3 Security and Usability

The process of allowing a principal to perform an action on the system can be broken up into *identification* ("what is your name?"), *verification* ("prove that you are the one you claim to be") and *authorization* ("based on who you are, I'll let you do this"). The first two steps are usually taken together as *authentication*.

We certainly won't be alone in claiming that traditional methods of authentication of users to computers, such as passwords, are not suitable, from a usability perspective, for ubiquitous computing environments. Our focus is to make the three steps above more usable, without compromising security.

But more usable for whom? Certainly for the guests themselves; but also, and this is probably the more substantial challenge, for their hosts, who are the ones responsible for the complex task of defining the authorization rules, as well as the ones who stand to lose the most in case of security failures.

1.4 Identification: Defining Guests

To implement a system where guests can perform certain actions we need to define what a guest is and distinguish them from strangers who happen to be in range of the system.

There are various properties that guests have in the real world. Detecting such properties directly would allow people to use a guest system in an intuitive manner. Firstly, and most simply, a guest is inside the house, whereas a stranger is outside. With a position-based smart home this would be relatively easy to determine; however, only knowing proximity would give rise to too many annoying false negatives and dangerous false positives on the boundaries.

Guests can also be distinguished in that they have the permission of the owner (or more generally, any authorized principal) to be there.

1.5 Authorization: Defining Access Permissions

Smart Homes will, in the future, be used by people who are not normally security conscious and do not want to spend any effort configuring a system for guests. Therefore, even if we can determine who is a guest, the way in which we grant them permissions must be one that a non-technical user can easily understand and operate.

Ease of use is one of the major challenges in this work. With existing models it is possible to define a security policy that expresses the desired rules; however, it would not be easily understood by the common man. A corollary to this is that normal use should not be made more difficult with the addition of guest support. While guests are fairly common, they are the exceptional case and not the common one and should not make day to day running of a smart home more complicated for hosts.

The easy way out is to have experts pre-configure a set of canned profiles and just give the naïve home owner a menu of such profiles to choose from, when authorizing a guest to use facilities in the home. But this does not solve the hard problem of actually giving hosts fine control over the permissions they grant, while still retaining usability. Using the simple-minded profile-based system, a host may be forced to assign a guest to a totally inappropriate profile if that is the only way that a certain necessary permission (*e.g.* changing the thermostat temperature in the guest's bedroom) can be granted.

2 Research Ideas

2.1 Mental Models and Social Expectations

If the ideas we present are to be acceptable to the general public they must be easy to use. One of the ways this can be achieved is by harnessing existing expectations about how appliances work by making new systems behave similarly to current systems.

In the case of most household appliances the security policy is that of the ‘Big Stick’ principle [10, §4.2.8]. That is, whoever has physical access to the device is allowed to control it. This is sufficient security because there are social restrictions on people’s actions. Guests are expected to behave in a certain fashion and there are social penalties which apply if they don’t.

We can harness these social expectations when moving to smart environments and in a lot of cases the Big Stick principle, combined with social restrictions, is still sufficient.

We can, of course, use the technology to improve on the current situation. Firstly, we can restrict more of the actions. While social restrictions may be enough to control some actions we may well want to enforce some of the more sensitive actions via technology. We can also provide improved logging to make it a lot more obvious when a social rule is being breached.

2.2 Mimicking the Big Stick

If we are dealing with a system that uses a wireless connection then merely having access to the control interface does not guarantee physical possession. We may not be a guest, but actually an attacker in the street.

A great variety of cryptographic key setup protocols have been developed to ensure that a secure channel is established between exactly two devices and that we can keep this channel on subsequent connections between the two devices. The problem is then reduced to identifying and verifying the devices correctly when they first connect.

Combining this with our desire to mimic the existing methods of controlling a device leads to the solution of requiring a physical button press on the device the first time an action is requested. This proves the presence of someone in the room and therefore their status as a guest. This is not quite enough, though; we need to confirm that the action which is confirmed is the same one that they requested and not an attacker. The use of multi-factor authentication [11] can solve this: for example the guest may be shown a nonce on the TV screen and asked to text the nonce back to the smart home via his mobile phone to prove that he could read it and therefore that he is inside.

If we have an integrated smart environment then this identification as a guest can be propagated across several devices.

2.3 Granting Permissions

The second part of this research is about how to give permissions to guests in an easy to manage way. In the traditional security model you would define permissions for all actions to each guest in advance, akin to setting the read, write and execute permission bits to appropriate values on all the files of your computer before letting a guest use it. However, this is not feasible or desirable when you don’t know about principals in advance, nor is it something that a non-trained user would be expected to do.

The first problem can be mitigated by giving the same permissions to all guests, but this is inflexible and still not something most users would be able to do. Another solution ("lazy evaluation" approach) could be to request approval for all actions by a guest from an administrator. This solves the problem of needing to do probably unnecessary work in advance, but gives a large penalty to actions done by guests—both for hosts, who are continuously interrupted with "can she do this?" requests, and for guests, who always have to wait until each action is individually authorized.

To mitigate these problems we suggest grouping actions in several ways. Firstly we have defined four types of action as related to guests:

1. Any guest may invoke without further authorization.
2. Authorization once for this action suffices for further invocations.
3. Each invocation requires individual authorization.
4. Guests may never invoke this.

The first category covers all functions for which the 'Big Stick' policy is still appropriate and we anticipate it will cover many of the functions available in household appliances. Category two contains functions which you might want to grant to some guests but not to others; however, once granted you are happy for those guests to access them as much as they like.

Using this grouping of functions allows appliance manufacturers to perform further grouping of security-equivalent functions. For example, on a smart Hi-Fi granting the play permission would always go hand in hand with the stop and pause permissions and so on. In the majority of cases, such functions would default to type 1.

For type 2 and 3 functions, invocation by a guest could cause a message to one of the known principals to authorize the request. This has several desirable properties. Firstly it allows the action to be authorized without any configuration in advance; and secondly it reinforces the social restriction on the action since the household member now knows what the guest is trying to do.

2.4 Ownership Delegation

In the situations described above we are giving permission to invoke functions whilst still retaining control of the device. There may be situations where we want to delegate control over a device, at least partially. For example, for guests staying for an extended period of time, we may wish to delegate the control of one room, including all the facilities and devices inside it, and the access control of that room or the house. We will ultimately want to retain full control over the room, but we might want to make some guarantees about how that control can be exercised so as to provide privacy to the guest.

3 Prototype System

We are building a prototype with the two purposes of trying out new ideas and then verifying whether they are as usable as we hope. The demo system shown at the workshop and described below was just a concept demo and therefore only explored the first point. As for the second point, we aim to develop a user study by deploying the system currently under development and letting non-technical users interact with it for some time. This aims to help us understand whether we are achieving an appropriate balance between security and usability.

3.1 Workshop Demo

The demo we presented at the workshop modelled a smart home environment containing two devices; A Hi-Fi and a display device such as a TV or a smart picture frame. The system is controlled through tokens (such as PDAs or mobile phones) carried by all the principals. The system was simulated using two laptops, one with the smart devices and the other showing the interface on the tokens.

This demonstrated the techniques of multi-factor authentication (having control of the PDA, being able to read the TV screen and being able to press a button on the Hi-Fi) to locate the guest inside the house. It also demonstrated the classification of functions according to their security relevance. Authorization for performing restricted functions was via a request to an identified administrator, caching this authorization for a period of time so that the permission did not need to be requested on subsequent invocations of that function.

3.2 Further Demos

We hope to expand this Demo into an environment which is closer to our target environment of a smart home and to involve users who are not members of our research group. Interviewing these users will give us valuable feedback about how the research should progress.

We are planning a deployment in which we can periodically evaluate the demo and then add features and improve the design based on the user feedback we collect. The evaluation will involve interviewing a selection of the users in the study to identify in which areas our design has succeeded and where it has failed. To this end we have started to design the questions we plan to use to collect useful feedback about our system design and on whether we are achieving a reasonable balance between security and usability.

Questions for Guests

- **Identification**
  - How inconvenienced are you by having to prove you are inside the home?
  - How easy was it to understand/do?
  - How easy/understandable was the multi-factor authentication? (depends on authentication method used)
- **Permissions**
  - Was it obvious what you could/couldn't do?
  - Was it easy to gain extra permissions?
  - Did you have to do this too often?

Questions for Administrators

- Did you have to grant anything you didn't want to?
- Was there anything you wanted to grant but couldn't?
- How easy was it to grant the correct permissions?
- Would you like to have done more in advance and less later?
- Would you like to have done less in advance and were happy to do more later?
- Were there areas where you would have liked to have finer grain control?
- Were there areas where the control was too complicated?

4 Conclusion

Smart Homes are on the verge of becoming a reality and, like all new systems, they will start off without much security. No satisfactory solutions to the problem of defining permissions for guests have been produced yet; but the issue needs to be solved, otherwise hosts will simply be forced to disable security features in order to accommodate their guests.

We aim to produce a solution that is flexible, easy to use and matches closely with the existing models of appliance use. We believe the correct balance between usability and security can only be reached through several iterations of user testing: we must allow non-experts to try out our ideas and tell us whether we got them right or not.

Acknowledgements

We are grateful to Alan Blackwell for precious advice about our future user study.

References

1. Kidd, C.D., Orr, R., Abowd, G.D., Atkeson, C.G., Essa, I.A., MacIntyre, B., Mynatt, E.D., Starner, T., Newstetter, W.: The aware home: A living laboratory for ubiquitous computing research. In: Streitz, N.A., Hartkopf, V. (eds.) CoBuild 1999. LNCS, vol. 1670, pp. 191–198. Springer, Heidelberg (1999)
2. Cooperstock, J.R., Tanikoshi, K., Beirne, G., Narine, T., Buxton, W.: Evolution of a reactive environment. In: Proc. of CHI 1995, Denver, CO, pp. 170–177 (1995)
3. Weatherall, J., Jones, A.: Ubiquitous networks and their applications. IEEE Wireless Communications 9, 18–19 (2002)
4. Sellen, A., Eardley, R., Izadi, S., Harper, R.: The whereabouts clock: early testing of a situated awareness device. In: CHI 2006: CHI 2006 extended abstracts on Human factors in computing systems, pp. 1307–1312. ACM Press, New York (2006)
5. Harper, R. (ed.): Inside the Smart Home. Springer, Heidelberg (2003)
6. IEEE 802.15 WPAN Task Group 4: ANSI/IEEE 802.15.4-2003, Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks. IEEE, New York (2003)

7. Hightower, J., Borriello, G.: Location systems for ubiquitous computing. IEEE Computer 34(8), 57–66 (2001)
8. Beresford, A.R.: Location privacy in ubiquitous computing. Technical Report UCAM-CL-TR-612, University of Cambridge, Computer Laboratory (2005)
9. Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. In: Mobile Computing and Networking, pp. 59–68 (1999)
10. Stajano, F.: Security for Ubiquitous Computing. John Wiley and Sons, Chichester (2002)
11. Wong, F.L., Stajano, F.: Multi-channel protocols. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2005. LNCS, vol. 4631, pp. 112–127. Springer, Heidelberg (2007)

Usability of Security Management: Defining the Permissions of Guests (Transcript of Discussion)

Matthew Johnson

University of Cambridge

George Danezis: So would you tell the user, the particular guest, that for some things you need authorisation, some things you don't?

Reply: If they don't get authorisation it will pop up and say, sorry, they didn't let you do this.

George Danezis: But if they do?

Reply: Then it will just happen.

George Danezis: Isn't it the case that for most guests these policies don't actually need active authorisation, but you just need logging, so if I'm invited to someone's house, it's pretty embarrassing if they find out that I've been looking through their collection of whatever?

Reply: Yes, it depends whether or not that's more embarrassing than having them finding something when they're looking through your collection of whatever.

Bruce Christianson: Like all your old episodes of Star Trek.

Reply: Yes. And there's other things which you probably don't want just to have logging for. So, it's nice if you can take an existing security policy that happens in the real world, and translate it into the digital world, because the existing security policy is probably good enough, and it's what people know, and understand, and for a lot of these things, guests aren't people who are complete strangers, so there is some sort of social restriction in place that you can tend to follow. Like George said, they'll be very embarrassed to be found looking through your personal papers.

Ross Anderson: But there are actually two interfaces on a typical client if you think about it, guests are normally implicitly defined as people who do not carry screwdrivers.

Reply: Yes. On the other hand that's also something which you don't expect: you would look askance at a guest who came and dismantled your hi-fi or whatever.

Bruce Christianson: It depends on the circles you move in.

Ross Anderson: But how these things do get managed in practice of course is that we have devices export their interfaces through remote controls, and so you have six or seven remote controls sitting on a coffee table?

Reply: Well the idea of some system like this is that you have one remote control which has all of all of the interfaces exported to it.

Bruce Christianson: Well you'd have several different remote controls...

Reply: Which all have all of them in fact, yes.

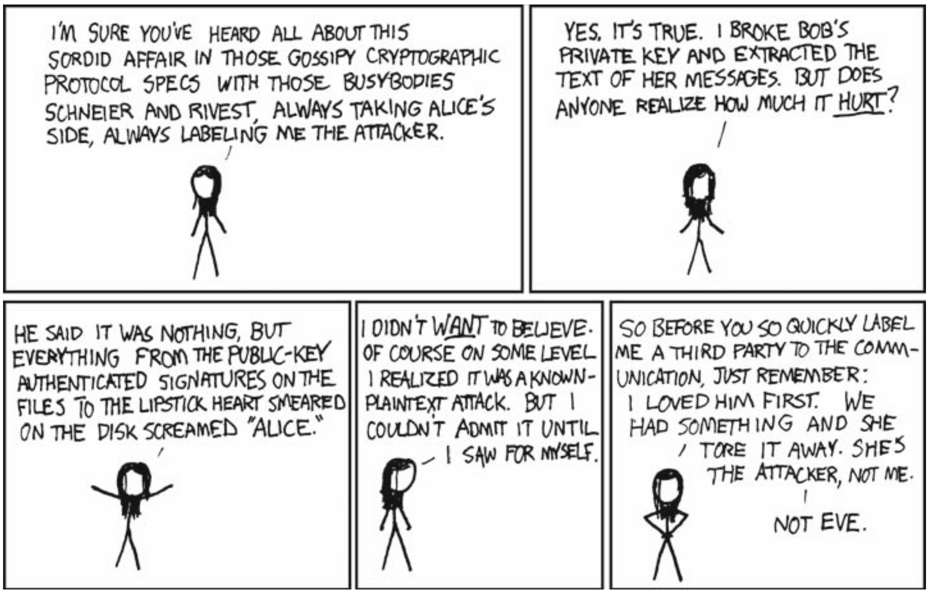
Frank Stajano: Saying that you'd like to export a request to a remote control doesn't address the problem, because you want to have different subsets of the functionality exported to the different remote controls held by people in different rooms.

Bruce Christianson: Oh, you're wanting to export the function to their own PDAs.

Frank Stajano: Yes, of course. When Matthew says remote control he means a universal remote control. They take on an interface, but the way that's exported to the remote control held by the guest, should not be the full administrative interface that's exported to the home owner. However the main point here is that, although I believe we know technically and cryptographically how to do delegation and all that stuff, we don't know how to do it in a way that would be acceptable to people who don't have a PhD in security.

The Last Word

Eve[*]


I'M SURE YOU'VE HEARD ALL ABOUT THIS SORDID AFFAIR IN THOSE GOSSIPY CRYPTOGRAPHIC PROTOCOL SPECS WITH THOSE BUSYBODIES SCHNEIER AND RIVEST, ALWAYS TAKING ALICE'S SIDE, ALWAYS LABELING ME THE ATTACKER.
YES, IT'S TRUE. I BROKE BOB'S PRIVATE KEY AND EXTRACTED THE TEXT OF HER MESSAGES. BUT DOES ANYONE REALIZE HOW MUCH IT HURT?
HE SAID IT WAS NOTHING, BUT EVERYTHING FROM THE PUBLIC-KEY AUTHENTICATED SIGNATURES ON THE FILES TO THE LIPSTICK HEART SMEARED ON THE DISK SCREAMED "ALICE."
I DIDN'T WANT TO BELIEVE. OF COURSE ON SOME LEVEL I REALIZED IT WAS A KNOWN-PLAINTEXT ATTACK. BUT I COULDN'T ADMIT IT UNTIL I SAW FOR MYSELF.
SO BEFORE YOU SO QUICKLY LABEL ME A THIRD PARTY TO THE COMMUNICATION, JUST REMEMBER: I LOVED HIM FIRST. WE HAD SOMETHING AND SHE TORE IT AWAY. SHE'S THE ATTACKER, NOT ME.
NOT EVE.

[*] Reproduced from `xkcd.com/177` by kind permission of Randall Munroe, who is *always* welcome to speak at SPW.

Author Index