The Correctness of Crypto Transaction Sets (Discussion)

Ross Anderson

Cambridge University

This talk follows on more from the talks by Larry Paulson and Giampaolo Bella that we had earlier. The problem I'm going to discuss is, what's the next problem to tackle once we've done crypto protocols? We keep on saying that crypto-protocols appear to be "done" and then some new application comes along to give us more targets to work on – multi-media, escrow, you name it. But sooner or later, it seems reasonable to assume, crypto will be done. What's the next thing to do?

The argument I'm going to make is that we now have to start looking at the interface between crypto and tamper-resistance.

Why do people use tamper resistance? I'm more or less (although not quite) excluding the implementation of tamper resistance that simply has a server sitting in a vault. Although that's functionally equivalent to many more portable kinds of tamper resistance, and although it's the traditional kind of tamper resistance in banking, it's got some extra syntax which becomes most clear when we consider the Regulation of Investigatory Powers (RIP) Bill. When people armed with decryption notices are going to be able to descend on your staff, grab keys, and forbid your staff from telling you, then having these staff working in a Tempest vault doesn't give the necessary protection.

1 Cryptography Involving Mutually Mistrustful Principals

In order to deploy "RIP-stop cryptography" (the phrase we've been using), you need to get guarantees which are mandatory – in the sense of mandatory access control. That is, you need guarantees, that are enforced independently of user action and are therefore subpoena proof, that a key was destroyed at date X, or that a key is not usable on packets over a week old, or that a key is only usable in packets over a week old if it's used with the thumb print of the corporate chief security manager, or whatever. You could do this with something like Kerberos, you just say that tickets can only be decrypted (as opposed to used) if the corporate security manager says so (this is maybe the cheap and cheerful way of getting RIP-stop into Windows). Alternatively, you could impose a requirement that a key should be unusable in the same key packet twice – possible if you've got a device like the 4758 with a reasonable amount of memory – or you could have a key which is unusable without a correct biometric. All of these give you ways of defeating RIP.

B. Christianson et al. (Eds.): Security Protocols, LNCS 2133, pp. 128–141, 2001.

Now when we look at the larger canvas, at the applications that really use hardware tamper-resistance, we find that most of them are used where there is mutual mistrust. Often all the principals in a system mistrust each other.

The classic example, that Simmons discussed in the 80's, was the business of nuclear treaty verification, and indeed command and control generally [10]. The Americans don't trust the Russians, and the Pentagon doesn't trust the commanders in the field not to use the weapons, and so you end up with this enormous hierarchy of tamper-resistant boxes.

Pre-payment electricity meters provide another application that we have discussed at previous workshops [3]. There you have a central electricity authority, and you have hundreds of regional electricity vendors who sell tokens which operate meters. How do you balance power and cash, and stop the various vendors running off with the cash – or selling tokens that they then don't own up to? The only known solution is using some kind of hardware tamper-resistance, at least in the vending stations.

Then you've got bank security modules, which are basically used because banks don't want to trust each others' programmers not to get at customer PINs. Matt asked yesterday: "How could you possibly use tamper-resistance where you have got mutual mistrust?" In practice, tamper-resistance is used precisely where there is mutual mistrust, and this is what makes it difficult and interesting.

A brief history of tamper-resistance includes weighted code-books, sigaba cypher machines with thermite charges, and so on – all fairly familiar. GSM is fairly low-level stuff: your SIM contains a customer key K_c, and this key is *you*, for practical purposes, in the network; it is used to respond to random authentication challenges. You have no real motive to break into the SIM; it's just convenient to use a smartcard to hold K_c (although it does prevent some kinds of cloning such as cloning a phone in a rental car).

Pay-TV becomes more serious, and Markus Kuhn has talked about it at some length [4]. There the mutual mistrust is between the pay-TV operator and the customer. The customer has the master key on his premises, and the pay-TV operator doesn't trust the customer not to use the master key. Hence the need for tamper-resistance.

Banking is an issue that I've written about a lot [2], and the kind of security modules that I worked with (and indeed built some of) are basically PCs in steel cases with lid switches. Open the lid, and bang goes the memory. This is a very simple way of building stuff. You can make it more complex by putting in seismometers, and photo-diodes, and all sorts of other sensors, but the basic idea is simple. When you take it to its logical conclusion you get the IBM 4758, where you have got a DES chip, microprocessor, battery and so on, potted in a tamper-sensing core with alarm circuits around it.

What are the vulnerabilities of such products?

Many of the kinds of vulnerability we had in the 80's were to do with maintenance access. The maintenance engineer would go in to change the battery; he could then disable the tamper-sensing; and he could then go back on his next

visit and take out the keys. That got fixed. The way to fix it was to take the batteries outside the device, as you can see in the photo of the 4758 in figure 1. Then there is nothing user-serviceable inside the device, so as soon as you penetrate the tamper-sensing membrane the thing dies irrevocably – it becomes a door-stop.

Fig. 1. – The IBM 4758 cryptoprocessor (courtesy of Steve Weingart)

We've more or less got to the point that the hardware can be trusted. We deal with the problem of whether the software can be trusted by getting FIPS 140-1 evaluation done by independent laboratories that sign a non-disclosure agreement and then go through the source code. That may be enough in some cases and not enough in others, but that isn't the subject of this talk.

2 Cryptoprocessor Transaction Sets

What I'm interested in is the command set that the cryptographic device itself uses. We have up till now, in this community, been looking at what happens in protocols where Alice and Bob exchange messages, and you need to contemplate only three or four possible types of message.

In the real world, things are much harder. If Alice and Bob are using tamper-resistant hardware, then they have got boxes that may support dozens or even

hundreds of transactions. And we are assuming that Alice and Bob are untrustworthy – perhaps not all the time, perhaps you're just worried about a virus taking over Alice's PC – but perhaps Alice is simultaneously your customer and your enemy, in which case you can expect a more or less continuous threat.

2.1 Early Transaction Sets – CUSP

Let me review quickly how cryptographic processor instruction sets developed. The first one that's widely documented is IBM's CUSP, which came in in 1980 in the PCF product, and shortly after that in the 3848 [8]. This was a device the size of a water-softener that hung off your mainframe on a channel cable. It was designed to do things like bulk file encryption, using keys that were protected in tamper-sensing memory.

What IBM wanted to do was to provide some useful protection for these keys. If you merely have a device which will, on demand, encrypt or decrypt with a key that is held in its memory, then it doesn't seem to do you much good. Anybody who can access the device can do the encrypting and the decrypting.

So there began to be a realization that we want, in some way or another, to limit what various keys can be used for. Although the details are slightly more complex than the subset I'm giving here, the idea that IBM had was that you can divide keys into different types, and some keys can be local keys and other keys can be remote keys. How they did this was to have three master keys for the device – basically one master key and two offsets that would be XORed with it to create dependent master-keys. A key that was encrypted with the main master key, you can think of that as a blue key, and a key that was encrypted with variant KMH_0 you might think of as a green key, and a key that was encrypted with variant KMH_1 you might think of as a red key.

Now few people go through this documentation [7], because the IBM terminology and notation in it are absolutely horrible. But by starting to think in terms of key typing, or key colours, you can start to make much progress.

One of the goals that IBM tried to achieve with the 3848 was to protect for host-terminal communications using a session key which is never available in the clear, and which therefore has no street value. (This was an NSA concern at the time, because of their various staff members who had been selling key material to the Russians.) So how do you protect communication between a mainframe and its terminal?

The implementation was that you had a host transaction ECPH, which would take a green key – that is a session key encyphered under KMH0 – and a message m, and it would, in the trusted hardware, get its hands on this green key by decyphering it under KMH0, and then apply it to encypher the message m.

ECPH: $\{KS\}_{KMHO}, m \longrightarrow \{m\}_{KS}$

So you had a means of using the green key to encypher messages, and 'can encypher with green key' is a property of CUSP. Similarly, you can decypher with

a green key, because can supply K_S encyphered under KMH_0 and a message encyphered under K_S and you will get back m.

How do you go about generating keys?

The technique is to supply a random number, or a serial number – or any value that you like, in fact – and use the device to decypher that under KMH0 to get a green key. In other words, the external encyphered keys that you hold on your database are just numbers that you thought up somehow. They are then 'decyphered' and used as keys.

Then you've got a more sensitive transaction, 'reformat to master key' (RFMK), which will take a master-key for a terminal – which is set up by an out-of-band technique – and K_S encrypted under KMH0, and it will encypher the session key under the master-key for the terminal.

RFMK: $\{KS\}_{KMHO}, KMT \longrightarrow \{KS\}_{KMT}$

So we've got another type of key, a red key I've called it, which is assumed to be held in the terminal. At the terminal, which is a different piece of hardware with different syntax, you've got a transaction DMK which will take a session key encyphered under KMT, and the clear value of KMT which is held in the hardware; it will give you a value of K_S which can then be used to do the decryption. (There isn't any hardware protection on the syntax of crypto in the terminal, because it is assumed that if you have physical access to the terminal then you can use the keys in it.)

Now this isn't all of the implementation, because there are other things that you need to do. You need to be able to manage keys, and (most confusingly of all) you need to be able to manage peer-to-peer communication between mainframes, which was being brought in at the time. So you end up having transactions to set up a key that is a local key at one mainframe and a remote key at the other mainframe. The IBM view at the time was that this made public-key cryptography unnecessary, because DES was so much faster and you needed tamper-resistant hardware anyway, so whay not just have local keys and remote keys? They have got the same functionality.

What went wrong with this? Well, it was very difficult to understand or explain, because the terminology they used is very complex. All sorts of implementation errors resulted. People found that it was simpler just to use blue keys which are all-powerful, rather than to mess around trying to separate keys into red and green and then finding that various things they wanted to do couldn't be done.

Nobody really figured out how to use this CUSP system to protect PINs properly as they were being sent from one bank to another. For example, a PIN from another bank's customer would come in from your cash machine, so you would decypher it and then re-encypher it with a key you shared with VISA for onward transmission. There were one or two hacks with which people tried to prevent PINs being available in clear in the host mainframe – such as treating PINs as keys – but for various reasons they didn't really work.

2.2 Banking Security Modules

So the next generation of security hardware to come along. The VISA security module, for example, takes the concept a bit further. They've actually got about five or six different colours of key.

The basic idea is that the security module is generating the PIN for a cash machine, by encrypting the account number using a key known as the PIN key. The result is then decimalised, and either given to the customer directly or added to an 'offset' on the customer's card to give the PIN that they must actually enter:

```
Account number:          8807012345691715
PIN key:                 FEFEFEFEFEFEFEFE
Result of DES:           A2CE126C69AEC82D
Result decimalised:      0224126269042823
Natural PIN:             0224
Offset:                  6565
Customer PIN:            6789
```

So you start off with keys which *never* decrypt, such as the PIN key KP, and you also have keys which *can* decrypt. Now the PIN key is a 'red key', a key which can never decrypt, and keys of this type are also used to protect PINs locally. (For example, when a PIN is encrypted at a cash machine for central verification, then the key used to perform this encryption is treated as a red key by the security module.) You can pass the security module a PIN which has been encrypted under a red key and it will say whether the PIN was right or not. The green key operates like a normal crypto key; you use it for computing MACs on messages and stuff like that.

So you use a red key to compute the PIN from the primary account number (PAN), and you use a red key to encipher the PIN from the ATM to the host security module. The whole thing works because nothing that's ever been encrypted under a red key is supposed to leak out – except the fraction of a bit per time that comes out from the PIN verification step. Then you add various support transactions, like 'generate terminal master key component', 'XOR two terminal master key components together to get a terminal master key' and so on and so forth.

What's the security policy enforced by the set of transactions that the security module provides? Well, it doesn't quite do Bell-LaPadula, because if you think of the red keys as being High, it allows an information flow from High to Low – about whether a PIN at High is right or not. So you need an external mechanism to track how many PIN retries have there been on an account, and if there's been more than three or 12 or whatever your limit is, then you freeze the account.

It doesn't quite do Clark-Wilson either, because you can generate master key components and XOR them together, and there's no system level protection for separation of duties on these two master key components; that's part of manual

procedure. But the security module is moving somewhat in the direction of Bell-LaPadula and Clark-Wilson (although its evolution went down a different path).

Once you network many banks together, you've got further requirements. You've got one type of key used to encrypt a PIN on the link between the cash machine and the security module of the acquiring bank; you've got another type of key being used between the acquiring bank and VISA; another type of key being used between VISA and the issuing bank; then you've got the top level master keys which VISA shares with banks, and so on. You can think of these as being all of different colours.

So you've got all these various types of key, and you've got more and more complex transactions; the VISA security module now has about 60 different transactions in its transaction set. You've got support transactions, such as translating different PIN formats. You also have got potential hacks which I will come to later.

So the concern with something like the VISA box is: "Is there some combination of these 60-odd transactions, which if issued in the right order will actually spit out a clear-text PIN?" The designers' goal was that there were supposed to be one or two – in order to generate PINs for customers and keys for ATMs, for example – but they all involve trusted transactions that involve entering a supervisor password or turning a metal key in a lock. So the issue of trusted subjects is supposedly tied down.

2.3 The IBM 4758 Security Module Product

Now we come to the more modern IBM product, the 4758. There's another picture of a 4758 in figure 2 with the shielding slightly removed. You can see a protective mesh here, a circuit board inside it behind a metal can, and the sensors here will alarm if you try to cut through (you can see the little sensor lines there). So what does this actually achieve? Well, assuming that the tamper-protection works (and it appears to), the software most people run on it – IBM's Common Cryptographic Architecture (CCA) – introduces a quite general concept of typing, namely control vectors.

A control vector can be thought of as a string bound to each key. The physical implementation of this is that you encipher the key under a different variant of a master key, but that's irrelevant at the level of abstraction that's useful to think about here. You can think of each key as going into the machine with a little tag attached to it saying, "I am a green key", "I am a red key", "I am a key of type 3126", or whatever. By default, CCA provides ten different types: there is a data key; there's a data translation key; there's a MAC key; there's a PIN generation key; there's an incoming PIN encryption key; and so on. There are somewhat over 100 different transactions supplied with CCA that operate using these key types. In addition, you can define your own key types and roll your own transactions. You can download the details from the Web, all 400 and whatever pages of it [7], so the target is public domain.

Even if you use the standard default types, there are some very obscure issues that are left to the designer. There are some possible hacks, or at least unclear

Fig. 2. – The 4758 partially opened – showing (from top left downwards) the circuitry, aluminium electromagnetic shielding, tamper sensing mesh and potting material (courtesy of Frank Stajano)

things, such as if people decrypt rubbish and use it, or if people use the wrong type of key, then can you gain some kind of advantage? Well keys supposedly have parity, so you have to decrypt on average about 32,000 'wrong' keys before you get one that works, but there have been attacks in the past which have used this [6]. Do any of these attacks work on the 4758?

Also, in the typing structure, we find that some of these types have got fairly explicit limitations (such as whether export is allowed or not), and you've got some types of types. But there's also a load of stuff that appears to be left more or less implicit, or at the very least has no supplied formal model.

You have got backward-compatibility modes. The 4758 will do everything that the old 3848 would do: it will do the ECPH, DCPH, re-format to master key, all that stuff, so you get free documentation of the obsolete equipment in the new equipment. You have things that approximate to the VISA transactions.

You also have a claim that the instruction set is comprehensive. Now this makes me feel rather uneasy, because the whole reason that I'm paying out all this money for a security processor is that its instruction set should *not* be comprehensive. There should be some things that it is simply not possible to do with it, and these things should be clearly understood.

So how did the current design come about? Well it should now be clear. We started off with the 3848, and that was not enough for banking so someone (actually Carl Campbell) invented the VISA security module. Then IBM saw

itself losing market share to the VSM, so it does the embrace-and-expand play and incorporates the VSM functionality in the 4753. This evolves into the 4758, which was then used for things like prepayment electricity meters [3]. People then invented i-buttons and smartcards, yet more applications with which IBM's product line had to be compatible. Roger's phrase, "the inevitable evolution of the Swiss army knife", is very appropriate.

3 Verifying Crypto Transaction Sets

So how can you be sure that there isn't some chain of 17 transactions which will leak a clear key? I've actually got some experience of this, because with a security module that I designed there was one particular banking customer who said, we need a transaction to translate PINs. They were upgrading all of their customers to longer account numbers, and they didn't want to change the customers' PINs because they were afraid that this would decrease the number of people using their credit cards in cash machines. So they said, "We'd like to supply two account numbers, an old account number and a new account number, plus a PIN, and you return the difference between the old PIN and the new PIN so we can write that as an offset onto the card track."

So I said, "Fine – we'll code that up. But be warned: this transaction is dangerous. Remove this version of the software as soon as you've done that batch job." Of course they didn't, and about a year and a half later one of their programmers realised: "Hey – if I put in my account number and the MD's account number and my PIN, the machine then tells me what I've got to subtract from my PIN in order to get the MD's PIN!" Luckily for the bank, he went to the boss and told all; to which our response was, "We told you, why didn't you do as you were told?" (This episode is described in [5].)

So managing this kind of thing isn't as straightforward as you might think. As I mentioned, we have got no proper formal model of what goes on in a device like this. Although it has some flavours of Bell-LaPadula and some flavours of Clark-Wilson, it doesn't really behave according to either model; there are leaks at the edges. This brings us to all the research opportunities, which aren't necessarily 'insurmountable' but are, I suspect, a step change more difficult than the attacks on protocols that people have being doing up to now.

The first opportunity is to find an attack on the 4758. Find some property of the manual which even IBM will be forced to admit was not something they had in mind when they designed it. Alternatively prove that it's secure (though given the history of provable security, that's perhaps just as damning). Or, more practically, provide a tool that enables the designer to use the 4758 safely. Up until now, the IBM philosophy appears to have been: "We're selling you an F15 and you had jolly well better let us sell you a pilot and a maintenance crew and all the rest of it at the same time." They are not alone in this respect; BBN and all the other vendors appear to take the same view.

So you find that you can go and buy a set of tamper-resistant hardware boxes for maybe $2,000 each, but if you are a bank and you are building an

actual installation using them, such as a certification authority, the outlay at the end of the day is more likely to be a quarter of a million dollars. By the time you've paid for the all the consultancy, and the machines they security modules will be attached to, and the firewalls to protect these machines from the network, and the procedures, and the training, and all the rest of it, the cost of the tamper-resistant box itself is essentially trivial.

So if anybody else is interested in getting into the systems integration business and getting that $240,000 instead of funneling it to IBM, then the competitive advantage might consist in having a suitable tool. Now it's not just the 4758! I use that as the target because we don't know how to break its physical tamper resistance, but there are many other architectures where you get a defined set of crypto transactions that may in fact be quite extensible. Those that come to mind are Microsoft CAPI, Intel CDSA – there's dozens out there.

This is definitely more difficult than doing verification of crypto protocols – and more interesting I think – because this is the real world problem. Somebody who has got a penetration of your corporate network is not trying to attack Kerberos, he's trying to attack the boxes that contain the Kerberos keys. Those boxes will do 50 transactions, of which you might normally use only three or four. But the others are there and available, unless you find some way of switching them off in a particular application, which again brings together a whole new set of design issues. So what's the nature of a crypto transaction set in general, and how do you go about verifying it?

Next question: how do you relate its properties to a higher level security policy model, such as Bell-LaPadula? And where this I suppose leads to, from the theory and semantics point of view, is whether you can get a general theory of restricted computation of computers that can do some things but cannot – absolutely, definitely, provably, cannot – do other things. Is there such a language that is complete in the sense that you can express any worthwhile crypto transaction set in it? What do you have to add to mathematics in order to get the expressiveness of metal? In the RIP-stop application it may very well be that provable destruction of a key is all you need, and anything else can be built on top of that given various external services such as perhaps secure time. But provable destruction of a key doesn't seem obviously related to what's going on in banking, because a typical bank is still running with the same ATM keys that they set up in 1985 when VISA set up the ACDS network. Key expiration just doesn't come into their universe.

And finally, if you're going to develop such a language, it would be useful if it could deal with public key stuff as well, such as homomorphic attacks, blinding, and so on. Because if you're going to build a machine which will, for example, not decrypt the same PGP message twice, then you have got somehow to make sure that the bad guys don't send you a message which is an old message which has been blinded so it looks like a new message.

Matt Blaze: I was surprised by your early comment – which I realise was a throwaway – that tamper-resistant hardware appears to be almost done, so let's move on to the next thing. It seems to me that tamper-resistant hardware

if anything has become increasingly tenuous of late, particularly with respect to the use of out-of-band attacks such as power analysis and timing attacks. Things have been made even more difficult by putting the power supply outside of the box. Do you think once Paul Kocher gets his hands on one of these IBM boxes and hooks up his spectrum analyser to it that that's the end?

Reply: In the case of the IBM product I don't think so, because they've got a reasonably convincing proof (in the electrical engineering sense) that this can't happen. They filter the power lines into the device in such a way that all the interesting signals, given the frequencies that they're at inside the box, are attenuated more than 100 dBs. I agree that there is a serious problem with smartcards, but we've got a project going on that, we believe it to be fixable. We discuss this in a paper at a VLSI conference recently [9]. What we're proposing to do is to use self-timed dual-rail logic, so that the current that is consumed by doing a computation is independent of the data that is being operated on. You can do various other things as well; you can see to it that no single transistor failure will result in an output, and so on. In fact the NSA already uses dual-rail for resilience rather than power-analysis reasons. So there are technologies coming along that can fix the power analysis problem.

When I say that hardware tamper-resistance has been done, what I'm actually saying is that we can see our way to products on the shelf in five or seven years time that will fix the kind of things that Paul Kocher has been talking about.

John Ioannidis: The kind of tamper-resistance you seem to analyse here is not the sort of tamper resistance I'm likely to find in my wallet or my key-ring. It's more like the sort of tamper-resistance you find in a big box somewhere. Maybe that big box is sitting on my desk, or under my desk, but it's not going to be sitting on my person.

Reply: Wrong. The logical complexity that I've described here is independent of the physical size of the device. You consider the kind of SIM card that's likely to appear, with added-on transactions to support an electronic purse, and you're already looking at the same scale of complexity as a VISA security module. You're looking at much more difficult management problems, because you have got independent threads of trust: the phone company trusts its K_c and the bank trusts its EMV keys, and how do these devices interact within the smartcard itself? It suddenly becomes very hairy and we don't at present have the tools to deal with it.

Virgil Gligor: How many 4758s did IBM sell, do you know what proportion were penetrated relative to NT or Windows 98, or any of the things that you have on your desk?

Reply: They're selling of the order of a couple of thousand a year, I'm aware of only one penetration which occurred – during testing as a result of a manufacturing defect.

I assume that an NT box, or for that matter a Linux box, can be penetrated by somebody who knows what he's doing, and so you must assume that the host

to which your security module is attached belongs to the enemy, even if only from time to time.

(Indistinct question)

Reply: It's useful in security to separate design assurance from implementation assurance and from strength of mechanism. Now the strength of mechanism of the tamper-resistance mechanisms in 4758s are very much higher than a smartcard, and a smartcard is very much higher than an NT server, but that's a separate lecture course.

What I'm concerned about here is the logical complexity of the set of commands which you issue to your tamper resistant device, be it a smartcard, or an NT server, or a 4758, because if you screwed up somehow in the design of those, then it doesn't matter how much steel and how many capacitors and how many seismometers and how many men with guns you've got. You're dead.

(Indistinct question)

Reply: We brought the designer of the 4758 over and we grilled him for a day, and we've looked at the things partially dismantled, and we've got one or two off-the-wall ideas about how you might conceivably, if you spent a million bucks building equipment, have a go at it. But we don't have any practical ideas that we could offer to a corporation and say, "Give us a million bucks and with a better than 50% chance we'll have the keys out in a year." The 4758 is hard!

John Ioannidis: What kind of primitives am I going to be offloading on to tamper-proof hardware so that no matter what kind of penetration my NT box has, there won't be an order for a billion dollars to be transferred from somewhere else to my account without my consent, so I can't get framed for fraud.

Reply: The sort of thing that hardware boxes are traditionally used for is to protect the PIN key, the key that generates all the PINs of the eleven million customers in your bank, because if a bad guy gets hold of that you've had it, you have to re-issue your cards and PINs. The sort of thing that you can use a 4758 for in a high-value corporate environment is to see to it that a transaction will only go out signed with the great seal of Barclays Bank (or whatever) provided – let's say – in the case of a $100m transaction, at least two managers of grade 4 or above have signed it using the signing keys and their smartcards. You can set up a system involving the 4758 and the smartcards which will distribute the keys, manage the certificates, and then will basically enforce your Clark-Wilson policy.

John Ioannidis: I'm not a grade 4 manager at Goldman Sachs, or whatever they're called, Suppose I'm a poor AT&T employee, and I have a box on my desk and I do my banking transactions over it, and I have somewhere a key given me by the bank, or I created my own, or both, and I have a tamper-resistant device which I want to use to secure my transaction. Now the question is, what kind of operations should that thing support, and what kind of interface to me, that does not go through NT or Linux, so that a transaction could not be made without my personal, physical approval – not my electronic approval. This is also a hard problem.

Reply: Agreed. But it's not the problem I'm addressing. Many people have talked about signature tablets that will display text; you put your thumb print, for example, to OK this text and this is supposed to give you a high degree of non-repudiation. But that's a different problem; it's not the problem that interests me in the context of this talk.

Michael Roe: What Ross is deliberately ignoring is the effect of successful tamper-resistance. If you have a very secure transaction in a very physically secure processor then the attacker is going to attack the software that goes in between the user interface and the secure box.

Reply: In this talk what I'm interested in is the command set that you use to drive a tamper resistant device and how you go about defining that in a way that's sensible – and assuring yourself that the design holds up. (I'm talking about design assurance not about the human computer interface aspects.)

(Indistinct question)

Reply: There have been attacks on early pay TV systems where there were protocol failures in the smartcard. Now if you believe Gemplus figures that by 2003 there will be 1.4 billion microprocessor cards sold a year (and as the biggest OEM they should know about that) then clearly it is important in engineering and industrial and economic terms, as well as being a problem that's directly related to protocol verification. That's the reason why I bring it to this audience. It is a protocol problem; but it's the next step up in difficulty. And it's sufficiently different that there's an awful lot of new research for people to do, and, hopefully, new theories to discover.

(Indistinct question)

Reply: I'm interested in Clark-Wilson-type objects and Bell-LaPadula-type objects, because that's where the real world is. The usual objection to Clark-Wilson is, "Where's the TCB?" Now as an example application, banking security modules are 90% of the way towards a Clark-Wilson TCB, so this should also be of scientific interest.

(Indistinct question)

Reply: An awful lot of design effort is believed to be about to go into multi-function smartcards, where you've got a SIM card sitting in your WAP device, and into which signed applets can be downloaded by your bank, by your insurance company, by your health care provider, etc.

(Indistinct question)

Reply: Now separability is not the only issue here, there's the correctness of the transactions that people can issue, and whether the applications themselves can be sabotaged by somebody just issuing the right commands in order.

(Indistinct question)

Reply: Even if you can download these applets safely, the applets will have to talk to each other in many cases for it to be useful. For example, if you're going to use your WAP device to pay for hospital treatment, by transferring money from your electronic purse to your health card, then that involves having interfaces that allow transactions between the two, and that's hard to do.

References

1. RJ Anderson, *'Security Engineering – a Guide to Building Dependable Distributed Systems'*, Wiley (2001) ISBN 0-471-38922-6
2. RJ Anderson, "Why Cryptosystems Fail" in *Communications of the ACM* vol 37 no 11 (November 1994) pp 32–40; earlier version at `http://www.cl.cam.ac.uk/users/rja14/wcf.html`
3. RJ Anderson, SJ Bezuidenhoudt, "On the Reliability of Electronic Payment Systems", in *IEEE Transactions on Software Engineering* vol 22 no 5 (May 1996) pp 294–301; `http://www.cl.cam.ac.uk/ftp/users/rja14/meters.ps.gz`
4. RJ Anderson, MG Kuhn, "Tamper Resistance – a Cautionary Note", in *Proceedings of the Second Usenix Workshop on Electronic Commerce* (Nov 96) pp 1–11; `http://www.cl.cam.ac.uk/users/rja14/tamper.html`
5. RJ Anderson, MG Kuhn, "Low Cost Attacks on Tamper Resistant Devices", in *Security Protocols – Proceedings of the 5th International Workshop* (1997) Springer LNCS vol 1361 pp 125–136
6. M Blaze, "Protocol Failure in the Escrowed Encryption Standard", in *Second ACM Conference on Computer and Communications Security*, 2–4 November 1994, Fairfax, Va; proceedings published by the ACM ISBN 0-89791-732-4, pp 59–67; at `http://www.crypto.com/papers/`
7. IBM, *'IBM 4758 PCI Cryptographic Coprocessor – CCA Basic Services Reference and Guide*, Release 1.31 for the IBM 4758-001, available through `http://www.ibm.com/security/cryptocards/`
8. CH Meyer, SM Matyas, *'Cryptography: A New Dimension in Computer Data Security'*, Wiley, 1982
9. SW Moore, RJ Anderson, MG Kuhn, "Improving Smartcard Security using Self-timed Circuit Technology", Fourth ACiD-WG Workshop, Grenoble, ISBN 2-913329-44-6, 2000; at `http://www.g3card.com/`
10. GJ Simmons, "How to Insure that Data Acquired to Verify Treaty Compliance are Trustworthy", GJ Simmons, *Proceedings of the IEEE* v 76 no 5 (1988)

Tamper Resistance - a Cautionary Note

Ross Anderson, Markus Kuhn

Cambridge University
Computer Laboratory
Pembroke Street
Cambridge CB2 3QG
England

This paper is also available in postscript and pdf formats.

Abstract

An increasing number of systems, from pay-TV to electronic purses, rely on the tamper resistance of smartcards and other security processors. We describe a number of attacks on such systems - some old, some new and some that are simply little known outside the chip testing community. We conclude that trusting tamper resistance is problematic; smartcards are broken routinely, and even a device that was described by a government signals agency as `the most secure processor generally available' turns out to be vulnerable. Designers of secure systems should consider the consequences with care.

This paper was published by the USENIX Association in *The Second USENIX Workshop on Electronic Commerce Proceedings*, Oakland, California, November 18-21, 1996, pp 1-11, ISBN 1-880446-83-9. It won the best paper award at that conference.

---

1 Tamperproofing of cryptographic equipment

Many early cryptographic systems had some protection against the seizure of key material. Naval code books were weighted; rotor machine setting sheets were printed using water soluble ink; and some one-time pads were printed on cellulose nitrate, so that they would burn rapidly if lit [Kah67].

But such mechanisms relied on the vigilance of the operator, and systems were often captured in surprise attacks. So cryptographic equipment designed in recent years has often relied on technical means to prevent tampering. An example is the VISA security module, commonly used in banks to generate and check the personal identification numbers (PINs) with which customers authenticate themselves at automatic teller machines. It is basically a safe containing a microcomputer that performs all the relevant cryptographic operations; the safe has lid switches and circuitry which interrupts power to memory, thus

erasing key material, when the lid is opened [VSM86]. The idea is to deny the bank's programmers access to customer PINs and the keys that protect them; so when a customer disputes a transaction, the bank can claim that the customer must have been responsible as no member of its staff had access to the PIN [And94].

Evaluating the level of tamper resistance offered by a given product is thus an interesting and important problem, but one which has been neglected by the security research community. One of the few recent articles that discuss the subject describes the design of the current range of IBM products and proposes the following taxonomy of attackers [ADD+91]:

Class I (clever outsiders):
They are often very intelligent but may have insufficient knowledge of the system. They may have access to only moderately sophisticated equipment. They often try to take advantage of an existing weakness in the system, rather than try to create one.

Class II (knowledgeable insiders):
They have substantial specialized technical education and experience. They have varying degrees of understanding of parts of the system but potential access to most of it. They often have highly sophisticated tools and instruments for analysis.

Class III (funded organisations):
They are able to assemble teams of specialists with related and complementary skills backed by great funding resources. They are capable of in-depth analysis of the system, designing sophisticated attacks, and using the most advanced analysis tools. They may use Class II adversaries as part of the attack team.

The critical question is always whether an opponent can obtain unsupervised access to the device [Mor94]. If the answer is no, then relatively simple measures may suffice. For example, the VISA security module is vulnerable to people with occasional access: a service engineer could easily disable the tamper protection circuitry on one of her visits, and extract key material on the next. But this is not considered to be a problem by banks, who typically keep security modules under observation in a computer room, and control service visits closely.

But in an increasing number of applications, the opponent can obtain completely unsupervised access, and not just to a single instance of the cryptographic equipment but to many of them. This is the case that most interests us: it includes pay-TV smartcards, prepayment meter tokens, remote locking devices for cars and SIM cards for GSM mobile phones [And95]. Many such systems are already the target of well funded attacks.

So in what follows, we will assume that all attackers can obtain several examples of the target equipment. We will also ignore tampering at the circuit board level (though this has caused losses, for example, with prepaid electricity meters [AB96]) and rather concentrate on attacks aimed at recovering crypto key material stored in smartcards and other chip-level security processors.

2 Breaking smartcards and microcontrollers

The typical smartcard consists of an 8-bit microprocessor with ROM, EEPROM and RAM, together with serial input and output, all in a single chip that is mounted on a plastic carrier. Key material is kept in the EEPROM.

Designers of EEPROM based devices face a problem: erasing the charge stored in the floating gate of a memory cell requires a relatively high voltage. If the attacker can remove this, then the information will be trapped.

Early smartcards received their programming voltage on a dedicated connection from the host interface. This led to attacks on pay-TV systems in which cards were initially enabled for all channels, and those channels for which the subscriber did not pay were deactivated by broadcast signals. By covering the programming voltage contact on their card with tape, or by clamping it inside the decoder using a diode, subscribers could prevent these signals affecting the card. They could then cancel their subscription without the vendor being able to cancel their service.

Some cards are still vulnerable to this kind of attack, and it gives rise to a sporadic failure mode of some card-based public telephone systems: telephones where the relevant contact is dirty or bent may fail to decrement any user's card. However, the cards used nowadays in pay-TV decoders generate the required 12 V from the normal 5 V power supply using an on-chip oscillator and diode/capacitor network. This can push up the cost of an attack, but does not make it impossible: large capacitors can be identified under a microscope and destroyed with lasers, ultrasonics or focused ion beams. A chip prepared in this way can be investigated at will without the risk of erasing the EEPROM.

So our task is to classify the various logical and physical attacks on security processors and get some idea of the cost involved.

2.1 Non-invasive attacks

Unusual voltages and temperatures can affect EEPROM write operations. For instance, for the PIC16C84 microcontroller, a trick has become widely known that involves raising VCC to VPP - 0.5 V during repeated write accesses to the security bit. This can often clear it without erasing the remaining memory.

For the DS5000 security processor, a short voltage drop sometimes released the security lock without erasing secret data. Processors like the 8752 that can be used with both internal and external memory but that limit the switch between them to resets have been read out using low voltages to toggle the mode without a reset. Low voltage can facilitate other attacks too: at least one card has an on-board analogue random number generator, used to manufacture cryptographic keys and nonces, which will produce an output of almost all 1's when the supply voltage is lowered slightly.

For these reasons, some security processors have sensors that cause a reset when voltage or other environmental conditions go out of range. But any kind of environmental alarm will cause some degradation in robustness. For example, one family of smartcard processors was manufactured with a circuit to detect low clock frequency and thus prevent single-stepping attacks. However, the wild fluctuations in clock frequency that frequently occur when a card is powered up and the supply circuit is stabilising, caused so many false alarms that the feature is no longer used by the card's operating system. Its use is left to the application programmer's discretion. Few of them bother; those who do try to use it discover the consequences for reliability. So many cards can be single-stepped with impunity.

For similar robustness reasons, the under-voltage and over-voltage detection circuitry in many devices will not react to transients. So fast signals of various kinds may reset the protection without destroying the protected information, and attacks of this kind are now known in the community for quite a number of devices.

Power and clock transients can also be used in some processors to affect the decoding and execution of individual instructions. Every transistor and its connection paths act like an RC element with a characteristic time delay; the maximum usable clock frequency of a processor is determined by the maximum delay among its elements. Similarly, every flip-flop has a characteristic time window (of a few picoseconds) during which it samples its input voltage and changes its output accordingly. This window can be anywhere inside the specified setup cycle of the flip-flop, but is quite fixed for an individual device at a given voltage and temperature.

So if we apply a clock glitch (a clock pulse much shorter than normal) or a power glitch (a rapid transient in supply voltage), this will affect only some transistors in the chip. By varying the parameters, the CPU can be made to execute a number of completely different wrong instructions, sometimes including instructions that are not even supported by the microcode. Although we do not know in advance which glitch will cause which wrong instruction in which chip, it can be fairly simple to conduct a systematic search.

A typical subroutine found in security processors is a loop that writes the contents of a limited memory range to the serial port:

```
1  b = answer_address
2  a = answer_length
3  if (a == 0) goto 8
4  transmit(*b)
5  b = b + 1
6  a = a - 1
7  goto 3
8  ...
```

We can look for a glitch that increases the program counter as usual but transforms either the conditional

jump in line 3 or the loop variable decrement in line 6 into something else.

Finding the right glitch means operating the card in a repeatable way. All signals sent to it have to arrive at exactly the same time after reset for every test run. Many glitches can be tested for every clock cycle, until one of them causes an extra byte to be sent to the serial port. Repeating it causes the loop to dump the remaining memory, which if we are lucky will include the keys we are looking for.

Output loops are just one target for glitch attacks. Others are checks of passwords, access rights and protocol responses, where corruption of a single instruction can defeat the protection. A possible software countermeasure might be to avoid single-point-of failure instructions. This was common enough in the old days of unreliable hardware: a senior Cambridge computer scientist recalls that in the 1950's a prudent system programmer was someone who, having masked off three bits, would verify that the result did not exceed seven!

Hardware countermeasures include independent internal clock generators that are only PLL synchronized with the external reference frequency.

2.2 Physical attacks

Physical attacks on some microcontrollers are almost trivial. For example, the lock bit of several devices with on-chip EPROM can be erased by focusing UV light on the security lock cell, which is located sufficiently far from the rest of memory.

Current smartcards are slightly harder to attack, but not very much harder. They generally have little to prevent direct access to the silicon; the marketing director of a smartcard vendor claimed that there was simply no demand from their users for anything really sophisticated [Mae94]. The most that appears to be done is a capacitive sensor to detect the continued presence of the passivation layer [RE95], or an optical sensor under an opaque coating [AndA]. Similar robustness considerations apply to these detectors as to the ones discussed above; they are often not used, and when they are, they are fairly easy to detect and avoid.

Anyway, the typical chip module consists of a thin plastic basis plate of about a square centimetre with conductive contact areas on both sides. One side is visible on the final card and makes contact with the card reader; the silicon die is glued to the other side, and connected using thin gold or aluminium bonding wires. The chip side of the plastic plate is then covered with epoxy resin. The resulting chip module is finally glued into the card, which is available in ISO credit card format, in miniature format for some GSM systems, or in the case of some prepayment electricity meter systems and pay-TV systems resembles a small plastic key.

Removing the chip is easy. First, we use a sharp knife or hand lathe to cut away the plastic behind the chip module until the epoxy resin becomes visible. Now we settle a few drops of fuming nitric acid

(>98% HNO_3) on the resin and wait a few minutes until some of it has dissolved (the process can be accelerated by heating up the acid with an infra-red radiator). Before the acid dissolves too much epoxy and gets solid, we wash acid and resin away by shaking the card in acetone. We repeat this procedure around five to ten times until the silicon surface of the die is fully exposed. The chip can then be washed and will be fully functional unless one of the bonding wires has been damaged.

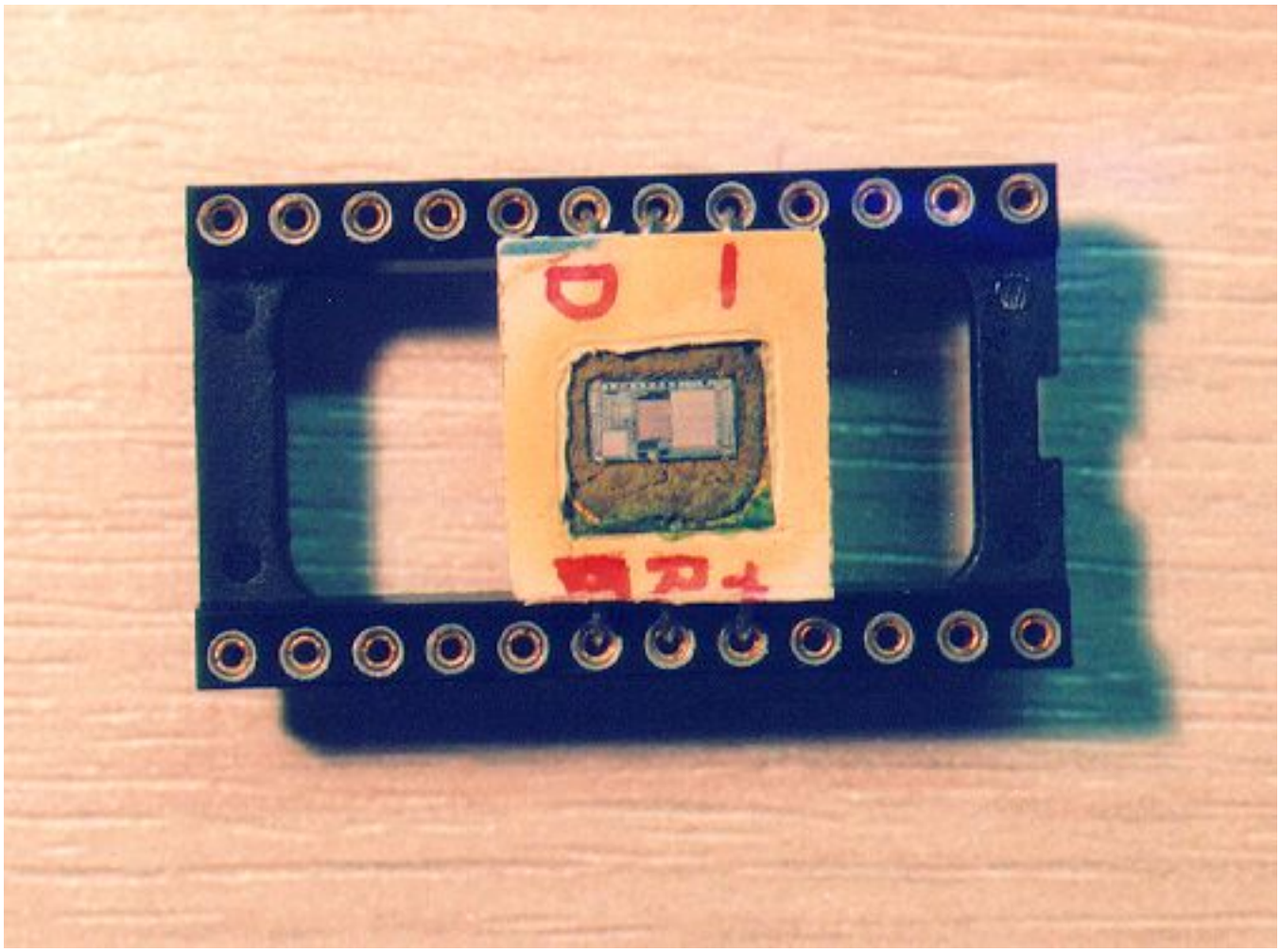

Figure 1: Fully functional smartcard processor with covering plastic removed for microprobing experiments. All tools necessary for this preparation were obtained for US$30 in a pharmacy.

Functional tests with pay-TV and prepaid phone smartcards have shown that EEPROM content is not affected by hot nitric acid. No knowledge beyond school chemistry is necessary; the materials are easily available in any chemistry lab, and several undergraduate students have recently reported the successful application of this method on an Internet mailing list dedicated to amateur smartcard hacking. Fuming nitric acid is an aggressive oxidant and should be handled carefully (especially when using flammable liquids such as acetone), but it does not affect silicon, silicon oxide, silicon nitride, or gold as used on the chip and its contacts. The aluminium used in the metal layer of the chip is covered at once with a thin

oxide layer and is also unaffected. Nitric acid is commonly used anyway to clean chip surfaces during manufacture.

There are commercial IC package removal machines used in process quality control, which expose the chip to an HNO_3 vapor stream that not only dissolves the resin but also transports away the waste products. This leaves a somewhat cleaner die surface than our manual method, but these machines use a lot of acid and need to be cleaned after use. So even professional chip analysis laboratories extract the chip manually if only a few packages have to be opened.

Most chips have a passivation layer of silicon nitride or oxide, which protects them from environmental influences and ion migration. This is not affected by nitric acid; chip testers typically remove it using dry etching with hydrogen fluoride, a process that is not as easily performed by amateur hackers.

But dry etching is not the only option. Another approach is to use microprobing needles that remove the passivation just below the probe contact point using ultrasonic vibration. Laser cutter microscopes commonly used in cellular biology laboratories have also been used to remove the passivation locally. Some testing laboratories have sets of nine microprobes so that the card bus can be read out during real time operation [BVR95].

It is also normal to remove the passivation before using an electron beam tester to access on-chip signals, because the secondary electrons emitted by the chip surface accumulate a positive charge on the passivation layer which causes the signals to disappear after a few seconds. One might therefore think that such attacks would require dry etching facilities. However, in some experiments with an electron beam tester, we have found that the charge accumulation effect is less serious when the chip is still covered with a thin dirt layer of HNO_3 and resin remains, which is probably weakly conductive. We suggest that a suitable weakly conductive layer might be deposited on top of the passivation layer as an alternative way of preventing the charge build-up.

2.3 Advanced attack techniques

The techniques described above have been successfully used by class I attackers - amateur pay-TV hackers, students and others with limited resources. We will now briefly describe some of the techniques available in professionally equipped semiconductor laboratories, of which there are several hundred worldwide. Some of these are situated in universities (three in the UK, for example), and it has happened that class I attackers get access to professional equipment in the course of student projects.

A recent article [BFL+93] gives an overview of a technique developed for reverse engineering chips at the Cavendish Laboratory in Cambridge. The authors of that paper first developed techniques for cleanly etching away a layer of a chip at a time. One innovation is a technique to show up N and P doped layers using the Schottky effect: a thin film of a metal such as gold or palladium is deposited on the chip creating a diode which can be seen with an electron beam. Images of successive layers of a chip are then

fed into a PC with image processing system software that reduces the initially fuzzy image to a clean polygon representation and identifies common chip features.

The system has been tested by reverse engineering the Intel 80386 and a number of other devices. The 80386 took two weeks, and it usually takes about six instances of a chip to get it right. The output can take the form of a mask diagram, a circuit diagram or even a list of the library cells from which the chip was constructed.

Once the layout and function of the chip are known, there is an extremely powerful technique developed by IBM for observing it in operation, without even having to remove the passivation layer. The tester places a crystal of lithium niobate over the feature whose voltage is to be monitored. The refractive index of this substance varies with the applied electric field, and the potential of the underlying silicon can be read out using an ultraviolet laser beam passed through the crystal at grazing incidence. The sensitivity of this technique is such that a 5 V signal of up to 25 MHz can be read [Wie90], and we understand that it is a standard way for well funded laboratories to recover crypto keys from chips of known layout. When attacking a smartcard, for example, we would read the EEPROM output amplifiers.

The response of the protection community to attacks of this kind has been to develop `conformeal glues', chip coatings that are not merely opaque and conductive but which also strongly resist attempts to remove them, usually damaging the underlying silicon in the process. These coatings are referred to in a FIPS standard [FIP94] and are widely used by the U.S. military, but are not generally available.

In addition to chip coatings, silicon features may be used to obscure the design. We have heard of design elements that look like a transistor, but are in reality only a connection between gate and source; and 3-input NORs which function only as 2-input NORs. Such copy traps may use holes in isolating layers or tricks done in the diffusion layer with ion implantation. However, the layer etching and Schottky techniques described above can detect them.

Another possibility is to introduce complexity into the chip layout and to use nonstandard cell libraries. However the chip still has to work, which limits the complexity; and nonstandard cells can be reconstructed at the gate level and incorporated in the recognition software.

A more systematic approach was employed in the U.S. government's Clipper chip. This had a fusible link system in which the links that created a classified encryption algorithm and a long term device key from an unclassified mask were fused after fabrication, and were made of amorphous silicon to make microscopy more difficult. In addition to this, the surface of the chip was `salted' with oscillators to make electromagnetic sensor attacks more complicated.

Details of the fusible link technology can be found in a paper in the relevant data book [GW93], and from the scanning electron micrographs there, it is clear that - given enough effort - the secret information can be recovered by sectioning the chip (this technique has been used by the Cambridge team on obscure features in other chips). We are reliably informed that at least one U.S. chipmaker reverse engineered the

Clipper chip shortly after its launch. However the attacks that discredited the Clipper chip used protocol failures rather than physical penetration [Bla94] - a topic to which we will return later.

Sectioning is not the only way to reverse engineer a chip whose surface is well protected. For example, a recently declassified technique invented at Sandia National Laboratories involves looking through the chip from the rear with an infra-red laser using a wavelength at which the silicon substrate is transparent. The photocurrents thus created allow probing the device's operation and identification of logic states of individual transistors [Ajl95].

The use of sectioning leads us to a more general, and relatively unexplored, topic - attacks that involve actively modifying the target chip rather than merely observing it passively. It is well known that active opponents can mount much more severe attacks on both cryptographic protocols and algorithms than passive opponents can, and the same turns out to be true when reverse engineering chips.

We understand, for example, that production attacks carried out by some pay-TV pirates involve the use of a focussed ion beam (FIB) workstation. This device can cut tracks in a chip's metallisation layer, and deposit new tracks or isolation layers. It can also implant ions to change the doping of an area of silicon, and it can even build vias to conductive structures in the lowest layers of the chip. These machines cost several million U.S. dollars, but low-budget attackers can rent time on them from various semiconductor companies.

Armed with such a tool, attacks on smartcards become much simpler and more powerful. A typical attack involves disconnecting almost all of the CPU from the bus, leaving only the EEPROM and a CPU component that can generate read accesses. For example, the program counter may be left connected in such a way that the memory locations will be accessed in order as the device is clocked (see Fig. 2). Once this has been done, the attacker needs only a single microprobing needle or electro-optical probe to read the entire EEPROM contents. This makes the program much easier to analyse than in passive attacks, which typically yield only an execution trace; it also avoids the considerable mechanical difficulties of keeping several probes simultaneously located on bus lines that are perhaps a micrometre wide.

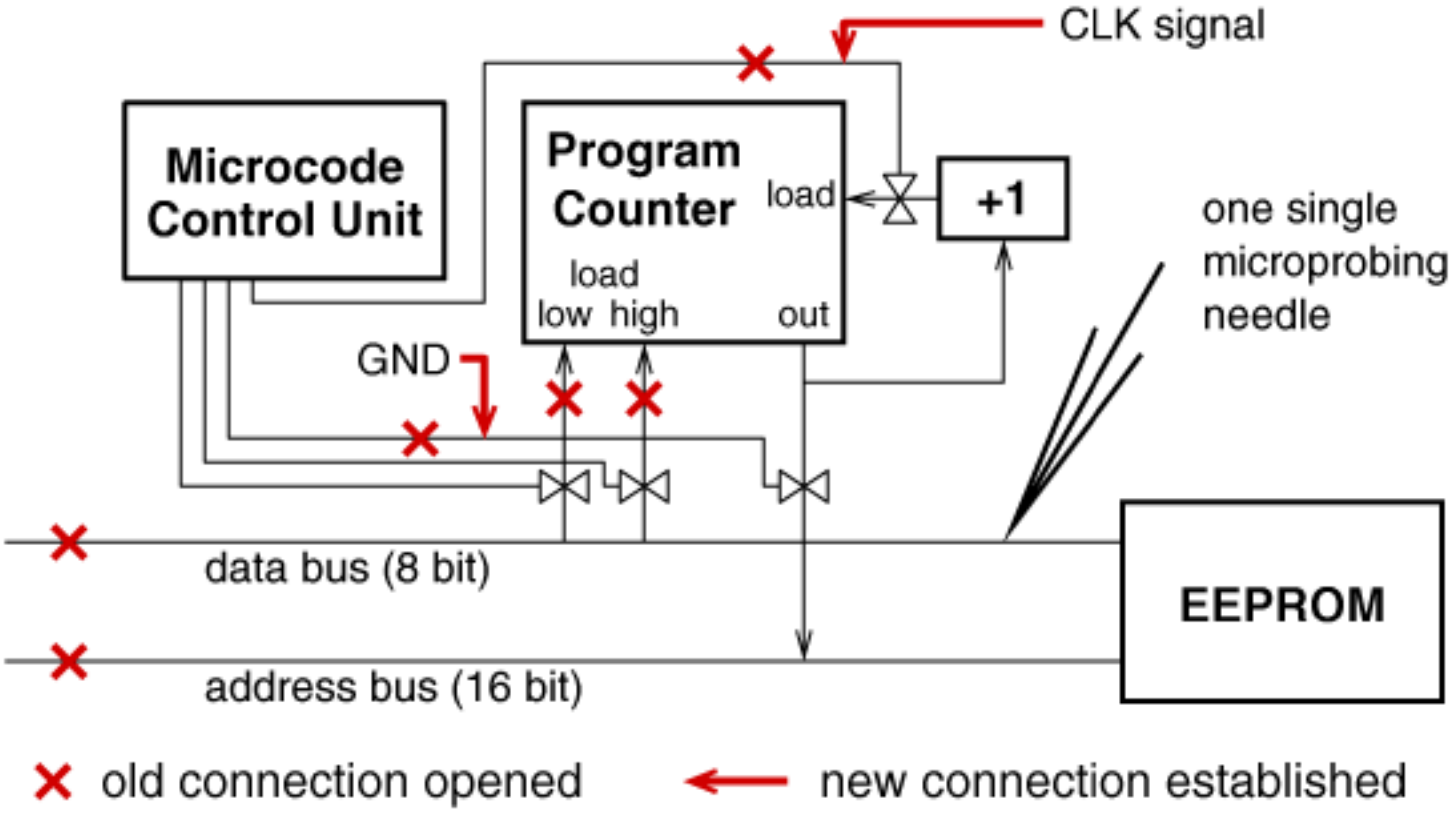

Figure 2: Read-out attack modifications on a security processor performed with a focused ion beam workstation allow easy access to secret EEPROM content with a single microprobing needle.

In conclusion, it is imprudent to assume that the design of silicon chips, or the information stored in them, can be kept from a capable motivated opponent. So how can we protect key material from such an opponent?

2.4 Advanced protection techniques

One application in which capable motivated opponents may be assumed, and where billions of dollars are spent on thwarting them, is the security of nuclear weapons. The threat model here is unequivocally class III - rogue states fronted by ``terrorist'' commando teams operating in cahoots with subverted military personnel. The U.S.A. has led the development of a control technology, now partially shared with other nuclear and near-nuclear nations, and the following account has been pieced together from a number of open sources.

Following the Cuban missile crisis, there was concern that a world war could start by accident - for example, by a local commander under pressure feeling that `if only they knew in Washington how bad things were here, they would let us use the bomb'. There was also concern that U.S. nuclear weapons in allied countries might be seized by the ally in time of tension, as U.S. forces there had only token custodial control. These worries were confirmed by three emergency studies carried out by Jerome

Wiesner, the presidential science adviser.

President Kennedy's response was National Security Action Memo no. 160, which ordered that America's 7,000 nuclear weapons then in countries from Turkey to Germany should be got under positive control, or got out [Sim93].

The U.S. Department of Energy was already working on safety devices for nuclear weapons, the basic principle being that a unique aspect of the environment had to be sensed before the weapon would arm. For example, missile warheads and some free-fall bombs expected to experience zero gravity, while artillery shells expected to experience an acceleration of 20,000 g. There was one exception though: atomic demolition munitions. These are designed to be taken from their storage depots to their targets by jeep or helicopter, or even hand carried by special forces, and then detonated using time fuses. So there is no scope for a unique environmental sensor to prevent accidental detonation.

The solution then under development was a secret arming code, which activated a solenoid safe lock buried deep in the plutonium pit at the heart of the weapon. The main engineering problem was that when the lock was exposed, for example by a maintenance engineer replacing the power supply, the code might become known (as with the VISA security module mentioned above). So it was not acceptable to have the same code in every weapon, and group codes had to be used; the same firing code would be shared by only a small batch of warheads.

But, following the Kennedy memo, it was proposed that all nuclear bombs should be protected using code locks, and that there should be a `universal unlock' action message that only the president or his legal successors could give. How could this be securely translated to a large number of individual firing codes, each of which would enable a small batch of weapons? The problem became worse when the Carter administration's policy of `measured response' created a need for a wide variety of `selective unlock' messages, giving the president options such as enabling the use of nuclear artillery and air defence weapons against a Soviet incursion into Germany. It became worse still with concern that a Soviet decapitation strike against the U.S. national command authority might leave the arsenal intact but useless. As is now well known, the solution lies in the branch of cryptomathematics known as `secret sharing' [Sch96], whose development it helped to inspire, and which enables weapons, commanders and options to be linked together with a complexity limited only by the available bandwidth.

In modern weapons the solenoid safe locks have been superseded by PALs - prescribed action links - about whose design details we have been able to find no useful open source material. However, it is known that PALs are considered sufficient only when they can be buried in the core of a large and complex weapon. With simple weapons (such as atomic demolition munitions) it is not considered feasible to deny access to a capable motivated opponent. These weapons are therefore stored in sensing containers called PAPS (prescribed action protective system) which provide an extra layer of protection.

Both the big-bomb and PAPS-enhanced systems include penalty mechanisms to deny a successful thief access to a usable weapon. These mechanisms vary from one weapon type to another but include gas

bottles to deform the pit and hydride the plutonium in it, shaped charges to destroy components such as neutron generators and the tritium boost, and asymmetric detonation that results in plutonium dispersal rather than yield. Whatever the combination of mechanisms used in a given design, it is always a priority to destroy the code in the switch; it is assumed that a renegade government prepared to deploy ``terrorists'' to steal a shipment of bombs would be prepared to sacrifice some of the bombs (and some technical personnel) to obtain a single serviceable weapon.

To perform authorised maintenance, the tamper protection must be disabled, and this requires a separate unlock code. The devices that hold the various unlock codes - for servicing and firing - are themselves protected in similar ways to the weapons. We understand, for example, that after tests showed that 1 mm chip fragments survived the protective detonation of a control device carried aboard airborne command posts, the software was rewritten so that all key material was stored as two separate components, which were kept at addresses more than 1 mm apart on the chip surface.

This highlights the level of care that must be taken when developing security processors that are to withstand capable attack. This care must extend to the details of implementation and operation. The weapons testing process includes not just independent verification and validation, but hostile `black hat' penetration attempts by competing laboratories or agencies. Even then, all practical measures are taken to prevent access by possible opponents. The devices (both munition and control) are defended in depth by armed forces; there are frequent zero-notice challenge inspections; and staff may be made to re-sit the relevant examinations at any time of the day or night.

These mechanisms and procedures have so far succeeded in preventing rogue governments from stealing (as opposed to making) atomic weapons.

The nuclear business also supplies the only examples known to us of tamper resistant packages designed to withstand a class III opponent who can obtain unsupervised physical access. These are the missile sensors developed to verify the SALT II treaty [Sim94] - which was never deployed - and the seismic sensor package developed for test ban treaty verification, which was. In this latter system, the seismic sensors are fitted in a steel tube and inserted into a drill hole that is backfilled with concrete. The whole assembly is so solid that the seismometers themselves can be relied upon to detect tampering events with a fairly high probability. This physical protection is reinforced by random challenge inspections.

So if systems have to be protected against class III opponents, we might hazard the following summary:

- if our goal is to merely detect tampering with a positive probability (as with treaty verification), then we can allow unsupervised access provided we are allowed to use a massive construction and to perform challenge inspections;
- if we wish to prevent the loss of a cryptographic key with near certainty (as with firing codes), then we had better use explosives and we had better also guard the device.

The above analysis convinced us that military agencies have limited confidence in the ability of tamper-

resistant devices (and especially portable ones) to withstand a class III opponent with unsupervised access. Having read an early draft of this paper, a senior agency official confirmed that chip contents cannot be kept from a capable motivated opponent; at most one can impose cost and delay. A similar opinion was ventured by a senior scientist at a leading chip maker.

Furthermore, the expense and inconvenience of the kind of protection used in the nuclear industry are orders of magnitude greater than even major banks would be prepared to tolerate. So what is the state of the art in commercial security processor design? They may be vulnerable to a class III opponent, but how about class II and class I?

3 Commercial security processors

Many commercial systems use either security module or smartcard technology. However, a growing number of designs consist of a composite package containing processor, memory, tamper detection circuitry and a battery.

An early example, whose design rationale was published in detail, is the µABYSS coprocessor developed by IBM. A variety of tamper resistant packages were tested for ease of penetration and ease of manufacturing, including stannic oxide lines on glass, piezo-electric sheets and a number of wire winding techniques. The designers settled on a four layer wrapping of 40 gauge (80 µm diameter) nichrome wire surrounding the processor, battery, memory and sensor circuitry, and embedded in a hard, opaque epoxy filled with silica to make it harder to machine and more likely to crack under UV laser ablation [WC87] [Wei87].

This appears to be a promising technology, and increasingly so as circuit sizes and power consumption shrink. The µABYSS design protected 260 cm^2 of card circuitry, but much less is used in many recent designs. 128 kilobyte SRAM chips are available today with room temperature data retention currents of less than 1 µA. A small 3 V lithium cell can easily provide this for a decade.

Many aggressive chemicals used to remove opaque chip packages (such as fuming nitric acid) have a low electrical resistance and can easily be detected as long as battery power is available; indeed, they will often cause critical breaks and short-circuits directly. Power supply networks could be made from a variety of different conductive and isolating materials such that practically any useful chemical solvent will cause at least one part to fail.

Suitable packaging can make it difficult for the attacker to strip away the protection one layer at a time, so that a successful attack might require a highly laborious process of manually shorting out the protective wire winding, guided by X-rays and precise measurements of the voltage at various points along its length.

There are some subtleties though. One might think that the protection mechanisms only have to

deactivate the power supply; but low-power SRAM chips remember bit values without a supply voltage at room temperature reliably for many seconds. By cooling the whole circuit with liquid nitrogen or helium, an attacker can extend the reliable power-off memory time to minutes or even hours, which could be enough to disable the alarm system and reapply power. Longterm exposure to a constant bit pattern can cause some SRAM cells to adapt their prefered power-up state accordingly, an effect that can remain for several days without any supply voltage [Gut96]. Possible countermeasures include SRAM cells with a well-defined power-up behavior.

Recent examples of battery-backed security module assemblies are the IBM Transaction Security System [ADD+91] and the Dallas Semiconductor DS5000 series [Dal93]. The latter devices have been described by a European signals security agency as the most secure processors available on general sale; we will now report a successful attack on them.

3.1 The Dallas DS5002FP Secure Microcontroller

One might want to make the tamper resistant module as small as possible, since hermetic sealing limits power dissipation, because larger packages are more vulnerable to false alarms, and for simple cost reasons.

But many applications require much more RAM than can be conveniently included in a small package, and one established technique is bus encryption [Bes79] [Bes80] [Bes81]. The CPU contains hardware for on-the-fly encryption of both the external address and the data bus. External RAM contains only encrypted data stored at encrypted addresses. The secret key is stored in a special battery buffered register on the CPU chip.

The Dallas Semiconductor DS5002FP microcontroller uses this bus encryption strategy. This Intel 8051 compatible processor is used in a number of financial transaction terminals and pay-TV access control systems to store secret keys and execute confidential cryptographic algorithms. On-chip bootloader firmware allows users to upload unencrypted software; it is then encrypted and stored in the external memory. The secret key is unique to each device, which has a special self-destruct pin that allows external alarms to erase it. A special version (DS5002FPM) features an additional metal layer die top coating designed to prevent microprobe attacks.

According to the manual, this layer is a ``*complex layout that is interwoven with power and ground which are in turn connected to logic for the Encryption Key and Security Logic. As a result, any attempt to remove the layer or probe through it will result in the erasure of the security lock and/or the loss of encryption key bits*''. Additional security is provided by pseudo-random dummy accesses performed on the external bus whenever the CPU core does not access external memory. In addition, 48 bytes including the reset and interrupt vectors are located on chip. Access to them also results in external dummy RAM access cycles, such that anyone observing the external bus cannot know when the internal RAM is accessed.

The security features of the DS5002FP are at first glance quite impressive and the manufacturer describes them as ``*the most sophisticated security features available in any microcontroller*''.

The chip uses two block ciphers that are loosely modelled on DES. The first encrypts addresses and acts on 15-bit blocks; the second encrypts data and acts on 8-bit blocks. The key of the second cipher is salted with the address of the byte being encrypted, but its small block size (which was no doubt dictated by the fact that the controller is byte oriented) turns out to be a useful feature for the attacker.

On closer examination, the algorithms show statistical weaknesses that might allow key recovery using differential cryptanalysis. We have not studied this in detail yet. In any case the algorithm strength is a purely economic issue; more rounds can buy more strength, but at a cost in either clock frequency or transistor count. Much more interesting is a weakness of the bus encryption system that is independent of the quality of the encryption algorithms.

3.2 Breaking the Dallas chip

One of us (Kuhn) has designed and demonstrated an effective practical attack that has already yielded all the secrets of some DS5002FP based systems used for pay-TV access control and also broken a code lock provided as a challenge by the German Federal Agency for Information Technology Security (BSI). The attack requires only a normal personal computer, a special read-out circuit built from standard electronic components for less than US$100, and a logic analyzer test clip for around US$200. It was performed in a student hardware laboratory at the University of Erlangen-Nürnberg using only common laboratory tools. Designing the hardware and software and performing the experiments leading to the final approach required less than three months. Thus, in the IBM taxonomy, this attack was carried out by a class I opponent.

The basic idea is simple, but was clearly not considered by the designers or evaluators of this processor. We call it the ``cipher instruction search attack'': it works by feeding the CPU with suitably chosen enciphered instructions and looking to see which of the resulting plaintext instructions we can recognise from their effects.

For example, we can recognise the three byte instruction

```
MOV 90h, #42h
```

encoded 75h 90h 42h, as it outputs byte value 42h on parallel port P1 (address 90h) two bus access cycles later.

So we reset the CPU and wait until some target instruction is about to be fetched. Then our read-out hardware replaces it, and our control software observes the reaction for a few more clock cycles. Then we repeat the procedure - which we can do over 300 times per second - and systematically work through all 2^{16} combinations for the first two encrypted instruction bytes.

We eventually find a two byte combination that sends a bijective function of the following byte to the parallel port. Assuming the first two bytes are the ciphertext corresponding to 75h 90h (which has to be confirmed by further tests), this gives the data bus decryption function at the address from which the third instruction byte was fetched. By testing all 2^8 values for this byte, we can tabulate the data decryption for one address.

Now we repeat the whole process. However this time we will search for a one-byte no-operation command (NOP) followed by the same MOV instruction as before. This effectively increases by one the address from which the third MOV instruction byte will be fetched.

Although we are now searching for a combination of four encrypted bytes representing two machine instructions, the search complexity has not been increased. We know already the correct encrypted value for one byte (port address 90h) from the previous tabulation of the encryption function at this address. The first instruction does not have to be a NOP, as any one-byte instruction that does not affect the following MOV will do. So the second search loop requires considerably less than 2^{16} iterations - in fact we typically need less than 2,500 attempts.

This search process becomes steadily faster as more addresses are tabulated, and we quickly tabulate the encryption function for a number of consecutive but still unknown addresses. We are now able to encrypt and send to the processor a sequence of machine instructions that simply dumps all the memory and special registers to one of the I/O ports.

The attack is in reality somewhat more complicated than presented in this brief overview. The details will be presented in a separate publication, together with a discussion of possible countermeasures for future bus encryption based systems. Our point is that a class I attacker could circumvent the physical protection of the `top' commercial system with modest effort. As the attack did not exploit either physical or cryptographic weaknesses, it might be considered a kind of protocol attack [AN95b].

4 Conclusion

It is prudent engineering practice to avoid single points of failure, and especially so where the likelihood of failure is unknown. This makes it all the more remarkable that the tamper resistance claims made for smartcards and other commercial security processors have gone untested for so long. The reader will by now be convinced that these claims should be treated with circumspection.

Public key techniques offer some solace, as the number of universal secrets can be greatly reduced - ideally, to a small number of certification keys, that can then be protected in depth. However, public key protocols have their own problems [AN95], and we should never forget that the great majority of actual security failures result from simple blunders in design, construction and operation [And94] [AB96]. There is no silver bullet.

A prudent engineer will see to it that the penetration of a moderate number of accessible devices, such as smartcards or payment terminals, will not be disastrous for the system. As most current electronic wallet systems use symmetric cryptography with universal secrets stored in retailers' terminals, they should be designed to keep on working after these secrets have been compromised - such as by supporting a fallback processing mode similar to that for credit cards, with full reconciliation, intrusion detection, hot lists and security recovery.

But although it is necessary to design commercial security systems with much more care, it is not sufficient. They must also be subjected to hostile testing. Readers may compare the nuclear weapons community's insistence on independent verification and validation with the aversion of the banking industry to any hostile review of their systems [And94b]. It is encouraging to note that some other sectors, such as the prepayment electricity meter industry, are starting to recognise the value of hostile review [AB96]. Other industries, such as pay-TV, got their hostile review once their systems were fielded.

Acknowledgements: This paper was largely written while the authors were guests at the Isaac Newton Institute, Cambridge. We also acknowledge useful comments from Bob Morris, Gus Simmons, Louis Guillou and a number of sources who prefer to remain anonymous; and support from the Fondazione Ugo Bordoni.

References

[ADD+91] DG Abraham, GM Dolan, GP Double, JV Stevens, ``Transaction Security System", in *IBM Systems Journal* v 30 no 2 (1991) pp 206-229

[Ajl95] C Ajluni, ``Two New Imaging Techniques Promise To Improve IC Defect Identification", in *Electronic Design* v 43 no 14 (10 July 1995) pp 37-38

[AndA] A Anderson <aha@apollo.HP.COM>, message posted to USENET sci.crypt 26 Apr 1994, message-ID <CovCG9.581@apollo.hp.com>

[And95] RJ Anderson, ``Crypto in Europe - Markets, Law and Policy" in *Cryptography: Policy and Algorithms*, Springer LNCS v 1029 pp 75-89

[And94b] RJ Anderson, ``Liability and Computer Security: Nine Principles", in *Computer Security - ESORICS 94*, Springer LNCS v 875 pp 231-245

[And94] RJ Anderson, ``Why Cryptosystems Fail", in *Communications of the ACM* v 37 no 11 (Nov 94) pp 32-40

[AB96] RJ Anderson, SJ Bezuidenhoudt, ``On the Reliability of Electronic Payment Systems'', in *IEEE Transactions on Software Engineering* v 22 no 5 (May 96) pp 294-301

[AN95b] RJ Anderson, RM Needham, ``Programming Satan's Computer'', in *Computer Science Today*, Springer LNCS v 1000 pp 426-441

[AN95] RJ Anderson, RM Needham, ``Robustness Principles for Public Key Protocols'', in *Advances in Cryptology - CRYPTO 95*, Springer LNCS v 963 pp 236-247

[Bla94] M Blaze, ``Protocol Failure in the Escrowed Encryption Standard'', in *Proceedings of the 2nd ACM Conference on Computer and Communications Security* (2-4 November 1994), ACM Press, pp 59-67

[BFL+93] S Blythe, B Fraboni, S Lall, H Ahmed, U de Riu, ``Layout Reconstruction of Complex Silicon Chips'', in *IEEE Journal of Solid-State Circuits* v 28 no 2 (Feb 93) pp 138-145

[BVR95] E Bovenlander, RL van Renesse, ``Smartcards and Biometrics: An Overview'', in *Computer Fraud and Security Bulletin* (Dec 95) pp 8-12

[Bes79] RM Best, ``Microprocessor for Executing Enciphered Programs'', U.S. Patent No. 4,168,396, September 18, 1979

[Bes80] RM Best, ``Preventing Software Piracy with Crypto-Microprocessors'', in *Proceedings of IEEE Spring COMPCON 80*, pp 466-469

[Bes81] RM Best, ``Crypto Microprocessor for Executing Enciphered Programs'', U.S. Patent No. 4,278,837, July 14, 1981.

[Dal93] *`Soft Microcontroller Data Book'*, Dallas Semiconductor, Dallas, Texas, 1993

[FIP94] *`Security Requirements for Cryptographic Modules'*, FIPS PUB 140-1, Federal Information Processing Standards Publication, National Institute of Standards and Technology, U.S. Department of Commerce, January 11, 1994

[GW93] KE Gordon, RJ Wong, ``Conducting Filament of the Programmed Metal Electrode Amorphous Silicon Antifuse'', in *Proceedings of International Electron Devices Meeting*, Dec 93; reprinted as pp 6-3 to 6-10, *QuickLogic Data Book* (1994)

[Gut96] P Gutmann, ``Secure Deletion of Data from Magnetic and Solid-State Memory'', in *Sixth USENIX Security Symposium Proceedings*, San Jose, California, July 22-25, 1996, pp 77-89

[Kah67] D Kahn, *`The Codebreakers'* (Macmillan 1967)

[Mae94] P Maes, marketing director, Gemplus, *comment during a panel discussion at Cardis 94*

[Mor94] R Morris, *talk given to Cambridge Protocols Workshop*, 1994

[RE95] W Rankl, W Effing, *`Handbuch der Chipkarten'*, Carl Hanser Verlag, 1995; ISBN 3-446-17993-3

[Sch96] B Schneier, *`Applied Cryptography - Protocols, Algorithms, and Source Code in C'* (second edition), John Wiley & Sons, New York, 1996

[Sim93] GJ Simmons, invited talk at the *1993 ACM Conference on Computer and Communications Security*, Fairfax, Virginia, Nov 3-5, 1993

[Sim94] GJ Simmons, ``Subliminal Channels; Past and Present'', *European Transactions on Telecommunications* v 5 no 4 (Jul/Aug 94) pp 459-473

[VSM86] *`VISA Security Module Operations Manual'*, VISA, 1986

[WC87] SR White, L Comerford, ``ABYSS: A Trusted Architecture for Software Protection'', in *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press pp 38-51

[Wei87] SH Weingart, ``Physical Security for the µABYSS System'', in *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, pp 52-58

[Wie90] JM Wiesenfeld, ``Electro-optic sampling of high-speed devices and integrated circuits'', in *IBM Journal of Research and Development* v 34 no 2/3 (Mar/May 1990) pp 141-161; *see also subsequent articles in the same issue*

Tamper Resistance — a Cautionary Note*

Ross Anderson

Cambridge University
Computer Laboratory
Pembroke Street
Cambridge CB2 3QG
England
rja14@cl.cam.ac.uk

Markus Kuhn

COAST Laboratory
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
U.S.A.
kuhn@cs.purdue.edu

Abstract

An increasing number of systems, from pay-TV to electronic purses, rely on the tamper resistance of smartcards and other security processors. We describe a number of attacks on such systems — some old, some new and some that are simply little known outside the chip testing community. We conclude that trusting tamper resistance is problematic; smartcards are broken routinely, and even a device that was described by a government signals agency as 'the most secure processor generally available' turns out to be vulnerable. Designers of secure systems should consider the consequences with care.

1 Tamperproofing of cryptographic equipment

Many early cryptographic systems had some protection against the seizure of key material. Naval code books were weighted; rotor machine setting sheets were printed using water soluble ink; and some one-time pads were printed on cellulose nitrate, so that they would burn rapidly if lit [20].

But such mechanisms relied on the vigilance of the operator, and systems were often captured in surprise attacks. So cryptographic equipment designed in recent years has often relied on technical means to prevent tampering. An example is the VISA security module, commonly used in banks to generate and check the personal identification numbers (PINs) with which customers authenticate themselves at automatic teller machines. It is basically a safe containing a microcomputer that performs all the relevant cryptographic operations; the safe has lid switches and circuitry which interrupts power to memory, thus erasing key material, when the lid is opened [27]. The idea is to deny the bank's programmers access to customer PINs and the keys that protect them; so when a customer disputes a transaction, the bank can claim that the customer must have been responsible as no member of its staff had access to the PIN [6].

Evaluating the level of tamper resistance offered by a given product is thus an interesting and important problem, but one which has been neglected by the security research community. One of the few recent articles that discuss the subject describes the design of the current range of IBM products and proposes the following taxonomy of attackers [1]:

Class I (clever outsiders): They are often very intelligent but may have insufficient knowledge of the system. They may have access to only moderately sophisticated equipment. They often try to take advantage of an existing weakness in the system, rather than try to create one.

Class II (knowledgeable insiders): They have substantial specialized technical education and experience. They have varying degrees of understanding of parts of the system but potential access to most of it. They often have highly sophisticated tools and instruments for analysis.

Class III (funded organisations): They are able to assemble teams of specialists with related and complementary skills backed by great funding resources. They are capable of in-depth analysis of the system, designing sophisticated attacks, and using the most advanced analysis tools. They may use Class II adversaries as part of the attack team.

*This paper has been published by the USENIX Association in *The Second USENIX Workshop on Electronic Commerce Proceedings*, Oakland, California, November 18–21, 1996, pp 1–11, ISBN 1-880446-83-9.

The critical question is always whether an opponent can obtain unsupervised access to the device [22]. If the answer is no, then relatively simple measures may suffice. For example, the VISA security module is vulnerable to people with occasional access: a service engineer could easily disable the tamper protection circuitry on one of her visits, and extract key material on the next. But this is not considered to be a problem by banks, who typically keep security modules under observation in a computer room, and control service visits closely.

But in an increasing number of applications, the opponent can obtain completely unsupervised access, and not just to a single instance of the cryptographic equipment but to many of them. This is the case that most interests us: it includes pay-TV smartcards, prepayment meter tokens, remote locking devices for cars and SIM cards for GSM mobile phones [4]. Many such systems are already the target of well funded attacks.

So in what follows, we will assume that all attackers can obtain several examples of the target equipment. We will also ignore tampering at the circuit board level (though this has caused losses, for example, with prepaid electricity meters [7]) and rather concentrate on attacks aimed at recovering crypto key material stored in smartcards and other chip-level security processors.

2 Breaking smartcards and microcontrollers

The typical smartcard consists of an 8-bit microprocessor with ROM, EEPROM and RAM, together with serial input and output, all in a single chip that is mounted on a plastic carrier. Key material is kept in the EEPROM.

Designers of EEPROM based devices face a problem: erasing the charge stored in the floating gate of a memory cell requires a relatively high voltage. If the attacker can remove this, then the information will be trapped.

Early smartcards received their programming voltage on a dedicated connection from the host interface. This led to attacks on pay-TV systems in which cards were initially enabled for all channels, and those channels for which the subscriber did not pay were deactivated by broadcast signals. By covering the programming voltage contact on their card with tape, or by clamping it inside the decoder using a diode, subscribers could prevent these signals affecting the card. They could then cancel their subscription without the vendor being able to cancel their service.

Some cards are still vulnerable to this kind of attack, and it gives rise to a sporadic failure mode of some card-based public telephone systems: telephones where the relevant contact is dirty or bent may fail to decrement any user's card. However, the cards used nowadays in pay-TV decoders generate the required 12 V from the normal 5 V power supply using an on-chip oscillator and diode/capacitor network. This can push up the cost of an attack, but does not make it impossible: large capacitors can be identified under a microscope and destroyed with lasers, ultrasonics or focused ion beams. A chip prepared in this way can be investigated at will without the risk of erasing the EEPROM.

So our task is to classify the various logical and physical attacks on security processors and get some idea of the cost involved.

2.1 Non-invasive attacks

Unusual voltages and temperatures can affect EEPROM write operations. For instance, for the PIC16C84 microcontroller, a trick has become widely known that involves raising VCC to VPP – 0.5 V during repeated write accesses to the security bit. This can often clear it without erasing the remaining memory.

For the DS5000 security processor, a short voltage drop sometimes released the security lock without erasing secret data. Processors like the 8752 that can be used with both internal and external memory but that limit the switch between them to resets have been read out using low voltages to toggle the mode without a reset. Low voltage can facilitate other attacks too: at least one card has an on-board analogue random number generator, used to manufacture cryptographic keys and nonces, which will produce an output of almost all 1's when the supply voltage is lowered slightly.

For these reasons, some security processors have sensors that cause a reset when voltage or other environmental conditions go out of range. But any kind of environmental alarm will cause some degradation in robustness. For example, one family of smartcard processors was manufactured with a circuit to detect low clock frequency and thus prevent single-stepping attacks. However, the wild fluctuations in clock frequency that frequently occur when a card is powered up and the supply circuit is stabilising, caused so many false alarms that the feature is no longer used by the card's operating system. Its use is left to the application programmer's discretion. Few of them bother; those who do try to use it discover the consequences for reliability. So many cards can be single-stepped with impunity.

For similar robustness reasons, the under-voltage and over-voltage detection circuitry in many devices will not react to transients. So fast signals of various kinds may reset the protection without destroying the protected information, and attacks of this kind are now known in the community for quite a number of devices.

Power and clock transients can also be used in some processors to affect the decoding and execution of individual instructions. Every transistor and its connection paths act like an RC element with a characteristic time delay; the maximum usable clock frequency of a processor is determined by the maximum delay among its elements. Similarly, every flip-flop has a characteristic time window (of a few picoseconds) during which it samples its input voltage and changes its output accordingly. This window can be anywhere inside the specified setup cycle of the flip-flop, but is quite fixed for an individual device at a given voltage and temperature.

So if we apply a clock glitch (a clock pulse much shorter than normal) or a power glitch (a rapid transient in supply voltage), this will affect only some transistors in the chip. By varying the parameters, the CPU can be made to execute a number of completely different wrong instructions, sometimes including instructions that are not even supported by the microcode. Although we do not know in advance which glitch will cause which wrong instruction in which chip, it can be fairly simple to conduct a systematic search.

A typical subroutine found in security processors is a loop that writes the contents of a limited memory range to the serial port:

```
1  b = answer_address
2  a = answer_length
3  if (a == 0) goto 8
4  transmit(*b)
5  b = b + 1
6  a = a - 1
7  goto 3
8  ...
```

We can look for a glitch that increases the program counter as usual but transforms either the conditional jump in line 3 or the loop variable decrement in line 6 into something else.

Finding the right glitch means operating the card in a repeatable way. All signals sent to it have to arrive at exactly the same time after reset for every test run. Many glitches can be tested for every clock cycle, until one of them causes an extra byte to be sent to the serial port. Repeating it causes the loop to dump the remaining memory, which if we are lucky will include the keys we are looking for.

Output loops are just one target for glitch attacks. Others are checks of passwords, access rights and protocol responses, where corruption of a single instruction can defeat the protection. A possible software countermeasure might be to avoid single-point-of failure instructions. This was common enough in the old days of unreliable hardware: a senior Cambridge computer scientist recalls that in the 1950's a prudent system programmer was someone who, having masked off three bits, would verify that the result did not exceed seven!

Hardware countermeasures include independent internal clock generators that are only PLL synchronized with the external reference frequency.

2.2 Physical attacks

Physical attacks on some microcontrollers are almost trivial. For example, the lock bit of several devices with on-chip EPROM can be erased by focusing UV light on the security lock cell, which is located sufficiently far from the rest of memory.

Current smartcards are slightly harder to attack, but not very much harder. They generally have little to prevent direct access to the silicon; the marketing director of a smartcard vendor claimed that there was simply no demand from their users for anything really sophisticated [21]. The most that appears to be done is a capacitive sensor to detect the continued presence of the passivation layer [23], or an optical sensor under an opaque coating [3]. Similar robustness considerations apply to these detectors as to the ones discussed above; they are often not used, and when they are, they are fairly easy to detect and avoid.

Anyway, the typical chip module consists of a thin plastic basis plate of about a square centimetre with conductive contact areas on both sides. One side is visible on the final card and makes contact with the card reader; the silicon die is glued to the other side, and connected using thin gold or aluminium bonding wires. The chip side of the plastic plate is then covered with epoxy resin. The resulting chip module is finally glued into the card, which is available in ISO credit card format, in miniature format for some GSM systems, or in the case of some prepayment electricity meter systems and pay-TV systems resembles a small plastic key.

Removing the chip is easy. First, we use a sharp knife or hand lathe to cut away the plastic behind

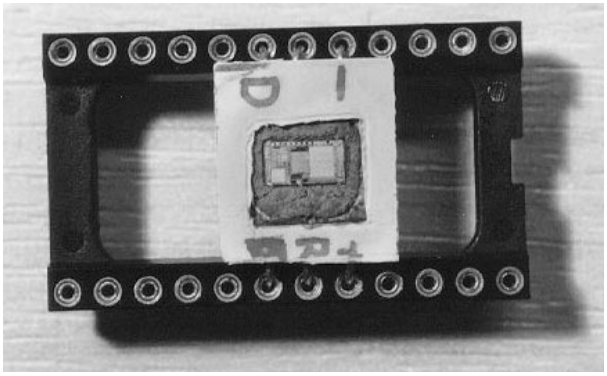

Figure 1: Fully functional smartcard processor with covering plastic removed for microprobing experiments. All tools necessary for this preparation were obtained for US$30 in a pharmacy.

the chip module until the epoxy resin becomes visible. Now we settle a few drops of fuming nitric acid (>98% HNO_3) on the resin and wait a few minutes until some of it has dissolved (the process can be accelerated by heating up the acid with an infra-red radiator). Before the acid dissolves too much epoxy and gets solid, we wash acid and resin away by shaking the card in acetone. We repeat this procedure around five to ten times until the silicon surface of the die is fully exposed. The chip can then be washed and will be fully functional unless one of the bonding wires has been damaged.

Functional tests with pay-TV and prepaid phone smartcards have shown that EEPROM content is not affected by hot nitric acid. No knowledge beyond school chemistry is necessary; the materials are easily available in any chemistry lab, and several undergraduate students have recently reported the successful application of this method on an Internet mailing list dedicated to amateur smartcard hacking. Fuming nitric acid is an aggressive oxidant and should be handled carefully (especially when using flammable liquids such as acetone), but it does not affect silicon, silicon oxide, silicon nitride, or gold as used on the chip and its contacts. The aluminium used in the metal layer of the chip is covered at once with a thin oxide layer and is also unaffected. Nitric acid is commonly used anyway to clean chip surfaces during manufacture.

There are commercial IC package removal machines used in process quality control, which expose the chip to an HNO_3 vapor stream that not only dissolves the resin but also transports away the waste products. This leaves a somewhat cleaner die surface than our manual method, but these machines use a lot of acid and need to be cleaned after use. So even professional chip analysis laboratories extract the chip manually if only a few packages have to be opened.

Most chips have a passivation layer of silicon nitride or oxide, which protects them from environmental influences and ion migration. This is not affected by nitric acid; chip testers typically remove it using dry etching with hydrogen fluoride, a process that is not as easily performed by amateur hackers.

But dry etching is not the only option. Another approach is to use microprobing needles that remove the passivation just below the probe contact point using ultrasonic vibration. Laser cutter microscopes commonly used in cellular biology laboratories have also been used to remove the passivation locally. Some testing laboratories have sets of nine microprobes so that the card bus can be read out during real time operation [12].

It is also normal to remove the passivation before using an electron beam tester to access on-chip signals, because the secondary electrons emitted by the chip surface accumulate a positive charge on the passivation layer which causes the signals to disappear after a few seconds. One might therefore think that such attacks would require dry etching facilities. However, in some experiments with an electron beam tester, we have found that the charge accumulation effect is less serious when the chip is still covered with a thin dirt layer of HNO_3 and resin remains, which is probably weakly conductive. We suggest that a suitable weakly conductive layer might be deposited on top of the passivation layer as an alternative way of preventing the charge build-up.

2.3 Advanced attack techniques

The techniques described above have been successfully used by class I attackers — amateur pay-TV hackers, students and others with limited resources. We will now briefly describe some of the techniques available in professionally equipped semiconductor laboratories, of which there are several hundred worldwide. Some of these are situated in universities (three in the UK, for example), and it has happened that class I attackers get access to professional equipment in the course of student projects.

A recent article [11] gives an overview of a technique developed for reverse engineering chips at the Cavendish Laboratory in Cambridge. The authors of that paper first developed techniques for cleanly etching away a layer of a chip at a time. One innovation is a technique to show up N and P doped layers using the Schottky effect: a thin film of a

metal such as gold or palladium is deposited on the chip creating a diode which can be seen with an electron beam. Images of successive layers of a chip are then fed into a PC with image processing system software that reduces the initially fuzzy image to a clean polygon representation and identifies common chip features.

The system has been tested by reverse engineering the Intel 80386 and a number of other devices. The 80386 took two weeks, and it usually takes about six instances of a chip to get it right. The output can take the form of a mask diagram, a circuit diagram or even a list of the library cells from which the chip was constructed.

Once the layout and function of the chip are known, there is an extremely powerful technique developed by IBM for observing it in operation, without even having to remove the passivation layer. The tester places a crystal of lithium niobate over the feature whose voltage is to be monitored. The refractive index of this substance varies with the applied electric field, and the potential of the underlying silicon can be read out using an ultraviolet laser beam passed through the crystal at grazing incidence. The sensitivity of this technique is such that a 5 V signal of up to 25 MHz can be read [30], and we understand that it is a standard way for well funded laboratories to recover crypto keys from chips of known layout. When attacking a smartcard, for example, we would read the EEPROM output amplifiers.

The response of the protection community to attacks of this kind has been to develop 'conformeal glues', chip coatings that are not merely opaque and conductive but which also strongly resist attempts to remove them, usually damaging the underlying silicon in the process. These coatings are referred to in a FIPS standard [17] and are widely used by the U.S. military, but are not generally available.

In addition to chip coatings, silicon features may be used to obscure the design. We have heard of design elements that look like a transistor, but are in reality only a connection between gate and source; and 3-input NORs which function only as 2-input NORs. Such copy traps may use holes in isolating layers or tricks done in the diffusion layer with ion implantation. However, the layer etching and Schottky techniques described above can detect them.

Another possibility is to introduce complexity into the chip layout and to use nonstandard cell libraries. However the chip still has to work, which limits the complexity; and nonstandard cells can be reconstructed at the gate level and incorporated in the recognition software.

A more systematic approach was employed in the U.S. government's Clipper chip. This had a fusible link system in which the links that created a classified encryption algorithm and a long term device key from an unclassified mask were fused after fabrication, and were made of amorphous silicon to make microscopy more difficult. In addition to this, the surface of the chip was 'salted' with oscillators to make electromagnetic sensor attacks more complicated.

Details of the fusible link technology can be found in a paper in the relevant data book [18], and from the scanning electron micrographs there, it is clear that — given enough effort — the secret information can be recovered by sectioning the chip (this technique has been used by the Cambridge team on obscure features in other chips). We are reliably informed that at least one U.S. chipmaker reverse engineered the Clipper chip shortly after its launch. However the attacks that discredited the Clipper chip used protocol failures rather than physical penetration [10] — a topic to which we will return later.

Sectioning is not the only way to reverse engineer a chip whose surface is well protected. For example, a recently declassified technique invented at Sandia National Laboratories involves looking through the chip from the rear with an infra-red laser using a wavelength at which the silicon substrate is transparent. The photocurrents thus created allow probing the device's operation and identification of logic states of individual transistors [2].

The use of sectioning leads us to a more general, and relatively unexplored, topic — attacks that involve actively modifying the target chip rather than merely observing it passively. It is well known that active opponents can mount much more severe attacks on both cryptographic protocols and algorithms than passive opponents can, and the same turns out to be true when reverse engineering chips.

We understand, for example, that production attacks carried out by some pay-TV pirates involve the use of a focussed ion beam (FIB) workstation. This device can cut tracks in a chip's metallisation layer, and deposit new tracks or isolation layers. It can also implant ions to change the doping of an area of silicon, and it can even build vias to conductive structures in the lowest layers of the chip. These machines cost several million U.S. dollars, but low-budget attackers can rent time on them from various semiconductor companies.

Armed with such a tool, attacks on smartcards become much simpler and more powerful. A typical attack involves disconnecting almost all of the

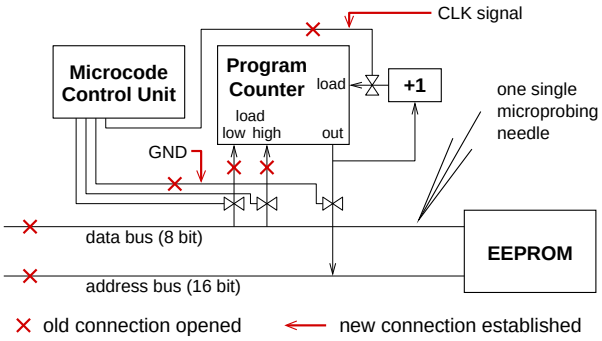

Figure 2: Read-out attack modifications on a security processor performed with a focused ion beam workstation allow easy access to secret EEPROM content with a single microprobing needle.

CPU from the bus, leaving only the EEPROM and a CPU component that can generate read accesses. For example, the program counter may be left connected in such a way that the memory locations will be accessed in order as the device is clocked (see Fig. 2). Once this has been done, the attacker needs only a single microprobing needle or electro-optical probe to read the entire EEPROM contents. This makes the program much easier to analyse than in passive attacks, which typically yield only an execution trace; it also avoids the considerable mechanical difficulties of keeping several probes simultaneously located on bus lines that are perhaps a micrometre wide.

In conclusion, it is imprudent to assume that the design of silicon chips, or the information stored in them, can be kept from a capable motivated opponent. So how can we protect key material from such an opponent?

2.4 Advanced protection techniques

One application in which capable motivated opponents may be assumed, and where billions of dollars are spent on thwarting them, is the security of nuclear weapons. The threat model here is unequivocally class III — rogue states fronted by "terrorist" commando teams operating in cahoots with subverted military personnel. The U.S.A. has led the development of a control technology, now partially shared with other nuclear and near-nuclear nations, and the following account has been pieced together from a number of open sources.

Following the Cuban missile crisis, there was concern that a world war could start by accident — for example, by a local commander under pressure feeling that 'if only they knew in Washington how bad things were here, they would let us use the bomb'. There was also concern that U.S. nuclear weapons in allied countries might be seized by the ally in time of tension, as U.S. forces there had only token custodial control. These worries were confirmed by three emergency studies carried out by Jerome Wiesner, the presidential science adviser.

President Kennedy's response was National Security Action Memo no. 160, which ordered that America's 7,000 nuclear weapons then in countries from Turkey to Germany should be got under positive control, or got out [25].

The U.S. Department of Energy was already working on safety devices for nuclear weapons, the basic principle being that a unique aspect of the environment had to be sensed before the weapon would arm. For example, missile warheads and some free-fall bombs expected to experience zero gravity, while artillery shells expected to experience an acceleration of 20,000 g. There was one exception though: atomic demolition munitions. These are designed to be taken from their storage depots to their targets by jeep or helicopter, or even hand carried by special forces, and then detonated using time fuses. So there is no scope for a unique environmental sensor to prevent accidental detonation.

The solution then under development was a secret arming code, which activated a solenoid safe lock buried deep in the plutonium pit at the heart of the weapon. The main engineering problem was that when the lock was exposed, for example by a maintenance engineer replacing the power supply, the code might become known (as with the VISA security module mentioned above). So it was not acceptable to have the same code in every weapon, and group codes had to be used; the same firing code would be shared by only a small batch of warheads.

But, following the Kennedy memo, it was proposed that all nuclear bombs should be protected using code locks, and that there should be a 'universal unlock' action message that only the president or his legal successors could give. How could this be securely translated to a large number of individual firing codes, each of which would enable a small batch of weapons? The problem became worse when the Carter administration's policy of 'measured response' created a need for a wide variety of 'selective unlock' messages, giving the president options such as enabling the use of nuclear artillery and air defence weapons against a Soviet incursion into Germany. It became worse still with concern that a Soviet decapitation strike against the U.S. national command authority might leave the arsenal intact but useless. As is now well known, the solution

lies in the branch of cryptomathematics known as 'secret sharing' [24], whose development it helped to inspire, and which enables weapons, commanders and options to be linked together with a complexity limited only by the available bandwidth.

In modern weapons the solenoid safe locks have been superseded by PALs — prescribed action links — about whose design details we have been able to find no useful open source material. However, it is known that PALs are considered sufficient only when they can be buried in the core of a large and complex weapon. With simple weapons (such as atomic demolition munitions) it is not considered feasible to deny access to a capable motivated opponent. These weapons are therefore stored in sensing containers called PAPS (prescribed action protective system) which provide an extra layer of protection.

Both the big-bomb and PAPS-enhanced systems include penalty mechanisms to deny a successful thief access to a usable weapon. These mechanisms vary from one weapon type to another but include gas bottles to deform the pit and hydride the plutonium in it, shaped charges to destroy components such as neutron generators and the tritium boost, and asymmetric detonation that results in plutonium dispersal rather than yield. Whatever the combination of mechanisms used in a given design, it is always a priority to destroy the code in the switch; it is assumed that a renegade government prepared to deploy "terrorists" to steal a shipment of bombs would be prepared to sacrifice some of the bombs (and some technical personnel) to obtain a single serviceable weapon.

To perform authorised maintenance, the tamper protection must be disabled, and this requires a separate unlock code. The devices that hold the various unlock codes — for servicing and firing — are themselves protected in similar ways to the weapons. We understand, for example, that after tests showed that 1 mm chip fragments survived the protective detonation of a control device carried aboard airborne command posts, the software was rewritten so that all key material was stored as two separate components, which were kept at addresses more than 1 mm apart on the chip surface.

This highlights the level of care that must be taken when developing security processors that are to withstand capable attack. This care must extend to the details of implementation and operation. The weapons testing process includes not just independent verification and validation, but hostile 'black hat' penetration attempts by competing laboratories or agencies. Even then, all practical measures are taken to prevent access by possible opponents. The devices (both munition and control) are defended in depth by armed forces; there are frequent zero-notice challenge inspections; and staff may be made to resit the relevant examinations at any time of the day or night.

These mechanisms and procedures have so far succeeded in preventing rogue governments from stealing (as opposed to making) atomic weapons.

The nuclear business also supplies the only examples known to us of tamper resistant packages designed to withstand a class III opponent who can obtain unsupervised physical access. These are the missile sensors developed to verify the SALT II treaty [26] — which was never deployed — and the seismic sensor package developed for test ban treaty verification, which was. In this latter system, the seismic sensors are fitted in a steel tube and inserted into a drill hole that is backfilled with concrete. The whole assembly is so solid that the seismometers themselves can be relied upon to detect tampering events with a fairly high probability. This physical protection is reinforced by random challenge inspections.

So if systems have to be protected against class III opponents, we might hazard the following summary:

- if our goal is to merely detect tampering with a positive probability (as with treaty verification), then we can allow unsupervised access provided we are allowed to use a massive construction and to perform challenge inspections;
- if we wish to prevent the loss of a cryptographic key with near certainty (as with firing codes), then we had better use explosives and we had better also guard the device.

The above analysis convinced us that military agencies have limited confidence in the ability of tamper-resistant devices (and especially portable ones) to withstand a class III opponent with unsupervised access. Having read an early draft of this paper, a senior agency official confirmed that chip contents cannot be kept from a capable motivated opponent; at most one can impose cost and delay. A similar opinion was ventured by a senior scientist at a leading chip maker.

Furthermore, the expense and inconvenience of the kind of protection used in the nuclear industry are orders of magnitude greater than even major banks would be prepared to tolerate. So what is the state of the art in commercial security processor design? They may be vulnerable to a class III opponent, but how about class II and class I?

3 Commercial security processors

Many commercial systems use either security module or smartcard technology. However, a growing number of designs consist of a composite package containing processor, memory, tamper detection circuitry and a battery.

An early example, whose design rationale was published in detail, is the μABYSS coprocessor developed by IBM. A variety of tamper resistant packages were tested for ease of penetration and ease of manufacturing, including stannic oxide lines on glass, piezo-electric sheets and a number of wire winding techniques. The designers settled on a four layer wrapping of 40 gauge (80 μm diameter) nichrome wire surrounding the processor, battery, memory and sensor circuitry, and embedded in a hard, opaque epoxy filled with silica to make it harder to machine and more likely to crack under UV laser ablation [28] [29].

This appears to be a promising technology, and increasingly so as circuit sizes and power consumption shrink. The μABYSS design protected 260 cm^2 of card circuitry, but much less is used in many recent designs. 128 kilobyte SRAM chips are available today with room temperature data retention currents of less than 1 μA. A small 3 V lithium cell can easily provide this for a decade.

Many aggressive chemicals used to remove opaque chip packages (such as fuming nitric acid) have a low electrical resistance and can easily be detected as long as battery power is available; indeed, they will often cause critical breaks and short-circuits directly. Power supply networks could be made from a variety of different conductive and isolating materials such that practically any useful chemical solvent will cause at least one part to fail.

Suitable packaging can make it difficult for the attacker to strip away the protection one layer at a time, so that a successful attack might require a highly laborious process of manually shorting out the protective wire winding, guided by X-rays and precise measurements of the voltage at various points along its length.

There are some subtleties though. One might think that the protection mechanisms only have to deactivate the power supply; but low-power SRAM chips remember bit values without a supply voltage at room temperature reliably for many seconds. By cooling the whole circuit with liquid nitrogen or helium, an attacker can extend the reliable power-off memory time to minutes or even hours, which could be enough to disable the alarm system and reapply power. Longterm exposure to a constant bit pattern can cause some SRAM cells to adapt their prefered power-up state accordingly, an effect that can remain for several days without any supply voltage [19]. Possible countermeasures include SRAM cells with a well-defined power-up behavior.

Recent examples of battery-backed security module assemblies are the IBM Transaction Security System [1] and the Dallas Semiconductor DS5000 series [16]. The latter devices have been described by a European signals security agency as the most secure processors available on general sale; we will now report a successful attack on them.

3.1 The Dallas DS5002FP Secure Microcontroller

One might want to make the tamper resistant module as small as possible, since hermetic sealing limits power dissipation, because larger packages are more vulnerable to false alarms, and for simple cost reasons.

But many applications require much more RAM than can be conveniently included in a small package, and one established technique is bus encryption [13] [14] [15]. The CPU contains hardware for on-the-fly encryption of both the external address and the data bus. External RAM contains only encrypted data stored at encrypted addresses. The secret key is stored in a special battery buffered register on the CPU chip.

The Dallas Semiconductor DS5002FP microcontroller uses this bus encryption strategy. This Intel 8051 compatible processor is used in a number of financial transaction terminals and pay-TV access control systems to store secret keys and execute confidential cryptographic algorithms. On-chip bootloader firmware allows users to upload unencrypted software; it is then encrypted and stored in the external memory. The secret key is unique to each device, which has a special self-destruct pin that allows external alarms to erase it. A special version (DS5002FPM) features an additional metal layer die top coating designed to prevent microprobe attacks.

According to the manual, this layer is a *"complex layout that is interwoven with power and ground which are in turn connected to logic for the Encryption Key and Security Logic. As a result, any attempt to remove the layer or probe through it will result in the erasure of the security lock and/or the loss of encryption key bits"*. Additional security is provided by pseudo-random dummy accesses performed on the external bus whenever the CPU core does not access external memory. In addition, 48 bytes including the reset and interrupt vectors are

located on chip. Access to them also results in external dummy RAM access cycles, such that anyone observing the external bus cannot know when the internal RAM is accessed.

The security features of the DS5002FP are at first glance quite impressive and the manufacturer describes them as *"the most sophisticated security features available in any microcontroller"*.

The chip uses two block ciphers that are loosely modelled on DES. The first encrypts addresses and acts on 15-bit blocks; the second encrypts data and acts on 8-bit blocks. The key of the second cipher is salted with the address of the byte being encrypted, but its small block size (which was no doubt dictated by the fact that the controller is byte oriented) turns out to be a useful feature for the attacker.

On closer examination, the algorithms show statistical weaknesses that might allow key recovery using differential cryptanalysis. We have not studied this in detail yet. In any case the algorithm strength is a purely economic issue; more rounds can buy more strength, but at a cost in either clock frequency or transistor count. Much more interesting is a weakness of the bus encryption system that is independent of the quality of the encryption algorithms.

3.2 Breaking the Dallas chip

One of us (Kuhn) has designed and demonstrated an effective practical attack that has already yielded all the secrets of some DS5002FP based systems used for pay-TV access control and also broken a code lock provided as a challenge by the German Federal Agency for Information Technology Security (BSI). The attack requires only a normal personal computer, a special read-out circuit built from standard electronic components for less than US$100, and a logic analyzer test clip for around US$200. It was performed in a student hardware laboratory at the University of Erlangen-Nürnberg using only common laboratory tools. Designing the hardware and software and performing the experiments leading to the final approach required less than three months. Thus, in the IBM taxonomy, this attack was carried out by a class I opponent.

The basic idea is simple, but was clearly not considered by the designers or evaluators of this processor. We call it the "cipher instruction search attack": it works by feeding the CPU with suitably chosen enciphered instructions and looking to see which of the resulting plaintext instructions we can recognise from their effects.

For example, we can recognise the three byte instruction

```
MOV 90h, #42h
```

encoded 75h 90h 42h, as it outputs byte value 42h on parallel port P1 (address 90h) two bus access cycles later.

So we reset the CPU and wait until some target instruction is about to be fetched. Then our read-out hardware replaces it, and our control software observes the reaction for a few more clock cycles. Then we repeat the procedure — which we can do over 300 times per second — and systematically work through all 2^{16} combinations for the first two encrypted instruction bytes.

We eventually find a two byte combination that sends a bijective function of the following byte to the parallel port. Assuming the first two bytes are the ciphertext corresponding to 75h 90h (which has to be confirmed by further tests), this gives the data bus decryption function at the address from which the third instruction byte was fetched. By testing all 2^8 values for this byte, we can tabulate the data decryption for one address.

Now we repeat the whole process. However this time we will search for a one-byte no-operation command (NOP) followed by the same MOV instruction as before. This effectively increases by one the address from which the third MOV instruction byte will be fetched.

Although we are now searching for a combination of four encrypted bytes representing two machine instructions, the search complexity has not been increased. We know already the correct encrypted value for one byte (port address 90h) from the previous tabulation of the encryption function at this address. The first instruction does not have to be a NOP, as any one-byte instruction that does not affect the following MOV will do. So the second search loop requires considerably less than 2^{16} iterations — in fact we typically need less than 2,500 attempts.

This search process becomes steadily faster as more addresses are tabulated, and we quickly tabulate the encryption function for a number of consecutive but still unknown addresses. We are now able to encrypt and send to the processor a sequence of machine instructions that simply dumps all the memory and special registers to one of the I/O ports.

The attack is in reality somewhat more complicated than presented in this brief overview. The details will be presented in a separate publication, together with a discussion of possible countermeasures for future bus encryption based systems. Our point is that a class I attacker could circumvent the

physical protection of the 'top' commercial system with modest effort. As the attack did not exploit either physical or cryptographic weaknesses, it might be considered a kind of protocol attack [8].

4 Conclusion

It is prudent engineering practice to avoid single points of failure, and especially so where the likelihood of failure is unknown. This makes it all the more remarkable that the tamper resistance claims made for smartcards and other commercial security processors have gone untested for so long. The reader will by now be convinced that these claims should be treated with circumspection.

Public key techniques offer some solace, as the number of universal secrets can be greatly reduced — ideally, to a small number of certification keys, that can then be protected in depth. However, public key protocols have their own problems [9], and we should never forget that the great majority of actual security failures result from simple blunders in design, construction and operation [6] [7]. There is no silver bullet.

A prudent engineer will see to it that the penetration of a moderate number of accessible devices, such as smartcards or payment terminals, will not be disastrous for the system. As most current electronic wallet systems use symmetric cryptography with universal secrets stored in retailers' terminals, they should be designed to keep on working after these secrets have been compromised — such as by supporting a fallback processing mode similar to that for credit cards, with full reconciliation, intrusion detection, hot lists and security recovery.

But although it is necessary to design commercial security systems with much more care, it is not sufficient. They must also be subjected to hostile testing. Readers may compare the nuclear weapons community's insistence on independent verification and validation with the aversion of the banking industry to any hostile review of their systems [5]. It is encouraging to note that some other sectors, such as the prepayment electricity meter industry, are starting to recognise the value of hostile review [7]. Other industries, such as pay-TV, got their hostile review once their systems were fielded.

Acknowledgements: This paper was largely written while the authors were guests at the Isaac Newton Institute, Cambridge. We also acknowledge useful comments from Bob Morris, Gus Simmons, Louis Guillou and a number of sources who prefer to remain anonymous; and support from the Fondazione Ugo Bordoni.

References

[1] DG Abraham, GM Dolan, GP Double, JV Stevens, "Transaction Security System", in *IBM Systems Journal* v 30 no 2 (1991) pp 206–229

[2] C Ajluni, "Two New Imaging Techniques Promise To Improve IC Defect Identification", in *Electronic Design* v 43 no 14 (10 July 1995) pp 37–38

[3] A Anderson `<aha@apollo.HP.COM>`, message posted to USENET `sci.crypt` 26 Apr 1994, message-ID `<CovCG9.581@apollo.hp.com>`

[4] RJ Anderson, "Crypto in Europe — Markets, Law and Policy" in *Cryptography: Policy and Algorithms*, Springer LNCS v 1029 pp 75–89

[5] RJ Anderson, "Liability and Computer Security: Nine Principles", in *Computer Security — ESORICS 94*, Springer LNCS v 875 pp 231–245

[6] RJ Anderson, "Why Cryptosystems Fail", in *Communications of the ACM* v 37 no 11 (Nov 94) pp 32–40

[7] RJ Anderson, SJ Bezuidenhoudt, "On the Reliability of Electronic Payment Systems", in *IEEE Transactions on Software Engineering* v 22 no 5 (May 96) pp 294–301

[8] RJ Anderson, RM Needham, "Programming Satan's Computer", in *Computer Science Today*, Springer LNCS v 1000 pp 426–441

[9] RJ Anderson, RM Needham, "Robustness Principles for Public Key Protocols", in *Advances in Cryptology — CRYPTO 95*, Springer LNCS v 963 pp 236–247

[10] M Blaze, "Protocol Failure in the Escrowed Encryption Standard", in *Proceedings of the 2nd ACM Conference on Computer and Communications Security* (2–4 November 1994), ACM Press, pp 59–67

[11] S Blythe, B Fraboni, S Lall, H Ahmed, U de Riu, "Layout Reconstruction of Complex Silicon Chips", in *IEEE Journal of Solid-State Circuits* v 28 no 2 (Feb 93) pp 138–145

[12] E Bovenlander, RL van Renesse, "Smartcards and Biometrics: An Overview", in *Computer Fraud and Security Bulletin* (Dec 95) pp 8–12

[13] RM Best, "Microprocessor for Executing Enciphered Programs", U.S. Patent No. 4,168,396, September 18, 1979

[14] RM Best, "Preventing Software Piracy with Crypto-Microprocessors", in *Proceedings of IEEE Spring COMPCON 80*, pp 466–469

[15] RM Best, "Crypto Microprocessor for Executing Enciphered Programs", U.S. Patent No. 4,278,837, July 14, 1981.

[16] *'Soft Microcontroller Data Book'*, Dallas Semiconductor, Dallas, Texas, 1993

[17] *'Security Requirements for Cryptographic Modules'*, FIPS PUB 140-1, Federal Information Processing Standards Publication, National Institute of Standards and Technology, U.S. Department of Commerce, January 11, 1994

[18] KE Gordon, RJ Wong, "Conducting Filament of the Programmed Metal Electrode Amorphous Silicon Antifuse", in *Proceedings of International Electron Devices Meeting*, Dec 93; reprinted as pp 6-3 to 6-10, *QuickLogic Data Book* (1994)

[19] P Gutmann, "Secure Deletion of Data from Magnetic and Solid-State Memory", in *Sixth USENIX Security Symposium Proceedings*, San Jose, California, July 22–25, 1996, pp 77–89

[20] D Kahn, *'The Codebreakers'* (Macmillan 1967)

[21] P Maes, marketing director, Gemplus, *comment during a panel discussion at Cardis 94*

[22] R Morris, *talk given to Cambridge Protocols Workshop*, 1994

[23] W Rankl, W Effing, *'Handbuch der Chipkarten'*, Carl Hanser Verlag, 1995; ISBN 3-446-17993-3

[24] B Schneier, *'Applied Cryptography – Protocols, Algorithms, and Source Code in C'* (second edition), John Wiley & Sons, New York, 1996

[25] GJ Simmons, invited talk at the *1993 ACM Conference on Computer and Communications Security*, Fairfax, Virginia, Nov 3–5, 1993

[26] GJ Simmons, "Subliminal Channels; Past and Present", *European Transactions on Telecommunications* v 5 no 4 (Jul/Aug 94) pp 459–473

[27] *'VISA Security Module Operations Manual'*, VISA, 1986

[28] SR White, L Comerford, "ABYSS: A Trusted Architecture for Software Protection", in *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press pp 38–51

[29] SH Weingart, "Physical Security for the μABYSS System", in *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, pp 52–58

[30] JM Wiesenfeld, "Electro-optic sampling of high-speed devices and integrated circuits", in *IBM Journal of Research and Development* v 34 no 2/3 (Mar/May 1990) pp 141–161; *see also subsequent articles in the same issue*