Smartcards - Tamperproofing and Known Attacks

Eric Auer (eric@coli.uni-sb.de)

June 19, 2000

1 *Abstract:*

For electronic commerce and secured (crypted/authenticated) communication you need a safe way to handle electronic keys and coins. If you store them in a small dedicated device like a smartcard you do not have to secure the complete system (eg. a PC used for home-banking). This makes security a lot easier, but smartcards have their own special weaknesses. This paper gives an introduction to common attacks on smartcards and on what can be done without too much effort to protect at least from low budget attacks.

2 *Introduction:*

The main reason to use smartcards for security is to keep unencrypted keys out of the probably insecure rest of your system. So the keys should not be extractable even under extreme conditions, for example if the attacker managed to steal the card.

The safelocks of nuclear weapons are buried deep inside the bombs and watched by armed guards. They even use explosives to destroy themselves if tampering attempts are detected, so an attacker will not gain access to the secrets stored inside.

Obviously you cannot protect your electronic car keys or the smartcard unlocking your pay-tv decoder to the same extent. In general, small devices do not even have their own power supply that could keep active security circuits for detecting attacks. So they cannot delete the keys under attack if tampering is detected.

I will discuss the following types of attacks along with a discussion of suitable countermeasures [AK96], [KK99]:

Non-invasive attacks:

- finding the internal state from external traces
- fault generation using unusual signals

Invasive attacks:

- reverse engineering
- eavesdropping with direct access to the silicon
- modifying circuits (eg. to ease eavesdropping)

3 *General attacks*

I will not discuss details of protocol and software-only attacks, as they are not specific to smartcards. An example of a protocol attack would be manipulating the lists of supported encryptions that Bob and Alice send to each other to establish a secure connection: If the attacker manages to remove all strong encryption methods, Bob and Alice will use weak encryption for their connection.

Sometimes an attacker just replaces your card with another one or modifies it to get access to the transmitted data. Therefore it is important to know that there are no modifications and the card is really yours. The former can be done with public key or challenge response protocols (to avoid giving away secret keys before the authentification is complete). For the "no modifications" condition, a card has to be tamper evident, ie. tampering must be detectable without too much effort. For the user it is useful to personalize the card by choosing a special design for the user interface, so simply replacing the card with another one from the same batch will not be enough to trick the user.

Because the user himself may be a potential attacker (eg. if the card stores keys functioning as electronic coins, the user will be tempted to make copies of his card) and attack tools can be hidden even inside a point of sale terminal, special precautions have to be taken. In general, a system based on smartcards must cope with the possibility that a few of the cards are cracked. It is a good thing to start with to allow revoking of cracked keys. And of course the encryption itself must withstand brute force and cryptanalytic attacks. See [AK96] for an attack on a processor using weak encryption for external memory accesses.

4 *Non-invasive attacks*

As those attacks do not require the device to be opened, they can often be done by modifying or faking a normal smartcard terminal. Even if tools are needed that cannot be hidden and the card must be stolen for analysis, there will usually be no further tamper evidence for the owner.

The more you know about the internal structure of a card the easier non-invasive attacks will be, so non-invasive attacks are often based on reverse engineering results or insider knowledge. For pay-tv keys an the like, the reverse engineering pays back quickly because non-invasive attacks scale well and can be applied to a large number of cards after development is complete.

4.1 *Eavesdropping*

4.1.1 *Attack description*

Eavesdropping is hard without access to internal bus lines. The best thing to do is analyzing the power consumption. Analyzing the electromagnetic field usually gives not enough accuracy. Every time a CMOS gate or a SRAM cell changes its state, there will be a short increase in power consumption, because a short circuit happens for a very short time during the switching. When a bus line changes its state, the bus drivers have to charge or discharge the load capacitance involved.

Measuring the current drawn by the device with high accuracy and a high sampling rate (compared to the CPU clock frequency), helps you to tell about the internal state of the CPU. You can assign a current peak to a change in a particular internal unit by measuring the delay between the clock edge and the current fluctuation. The delay differs between the different units of the chip.

As some crypto algorithms involve shifting key bits through flags, key bits can be recovered by eavesdropping processor flags. This can also be used to detect the outcome of some security check without having to wait for some message about the check from the card. In rare cases, even the algorithm used by a device can be deduced from its power consumption pattern.

4.1.2 *Countermeasures*

A strong countermeasure can be to do detailed analog simulations during development and to obscure recognizeable current peaks. Current regulators are useful for that. You can also randomize the execution by inserting idle cycles on a random basis so an attacker cannot associate a peak with the execution or outcome of a certain instruction. It is good to use an internal clock source for the same reason and to do only loose coupling to the external bus clock. Because the attacker wants to know the delay from clock edge to current peak, the attack will be harder if the clock signal is not visible from outside. An internal clock source will also protect from attacks involving low clock frequency to simplify measuring.

For the randomized execution you have to take care that the idle cycles exhibit the same activity patterns to an eavesdropper as normal cycles. Filter all incoming signals and supplies to keep them in a certain range, because non-standard conditions could make your random execution less random and may decrease the security of your device in various other ways.

4.2 *Fault generation*

4.2.1 *Attack description*

A simple example of fault generation is depriving a smartcard from enough power to write part of its non-volatile memory, for example to block it from changing the list of pay-tv channels it will unlock. If this succeeds, you can simply cancel your subscription to the vendor and keep on watching tv. Similar attacks were done on telephone cards. Variations in clock and power supply may cause random number generators to produce only weak randomness and tampering sensors to get less sensitive.

A wide range of attacks is based on *glitches* in signals and power. The idea is as follows: The capacitance and resistance of the parts of a chip determine certain signal propagation delays. Those delays show significant variance over the different units of the chip, but stay as they are if factors such as supply voltage and temperature are fixed.

So once you figure out the delays for a given unit of a given chip, you can apply very short glitches on signals that will only affect the state of a few flipflops. The reason is that flipflops sample their input only during a short time window relative to when the clock edge reaches them. A glitch at the right time on power supply will have similar effect, because the flipflop compares its input to the supply voltage for sampling.

It is more common and easier not to apply glitches to bus lines or power supply but to the clock input: A single clock cycle is made shorter so that some units sample their input before the updated state from the other units reaches them.

The goal of all those carefully designed faulty conditions is to change the flow of execution or some intermediate results. This can be used to ease eavesdropping or to bypass security checks. For example you can try to have the device skip a conditional jump that would be taken if authorization failed. Or change an encryption loop into a single round variant that is easier to decrypt.

Another good example are output loops: Imagine this:

i = 0
j = Number of Bytes to output
while j is above 0 do
(output Byte i of some Array, increase i, decrease j)

If you can change the comparison, its associated conditional jump or the decrement of j into something else, possibly secret data after the Array will be output. The fault must be reproduced for each round of the loop to have the loop never finish and output as much data as possible past the array.

4.2.2 *Countermeasures*

A method to protect a device is to use a segmented memory model. Each pointer (even the instruction pointer) is divided into a base and an offset. If the register hardware for the offset only allows few bits (e.g. 7), an attack similar to the example above may never reveal more than a few bytes. In addition, a wraparound in the offset can cause an immediate processor reset or hardcoded deletion of key data. This is useful to prevent invasive eavesdropping attacks using the instruction pointer, which are discussed below in section 5.5.

In general you should filter any connection between the card and the outside world to enforce sanity of all signals. The problem is that glitches are very short, so simple filters often will not be enough. Be careful that even a combination of strange signals will not get one of them through the filter. For the clock signal, it is best to generate your own right on the chip and do only loose coupling to the external bus clock. The power supply needed for updating non-volatile memory can be generated from the normal power supply using a charge pump on the chip.

A way to improve security in software is to check important things more than once like in this example:

y = x and 7, if y is below 8 then (do the rest)

To get y to grow above 7 you must sabotage two instructions, so guessing the right glitch will be much harder.

It is important to notice that hardware-only security is more important than software tricks here. If a sensor tells you that somebody tried to induce a fault, but your software fails to check the sensor at adequately short intervals, your sensor will be worthless. Some programmers just ignore sensor output to avoid false alarms.

5 Invasive attacks

Larger security modules can protect themselves better because they can contain their own power supply and complex sensors for intrusion detection. They can store key data in SRAM and delete the contents when a sensor alarm is triggered. A simple but not completely secure way is to cut power from the SRAM in that case, but part of the contents can be reconstructed if chip temperature is low and the attacker is fast enough. As smartcards are too small for even a small battery, they have to rely on passive protection while switched off. An attacker may disable intrusion sensors and then switch the device back on for further investigation.

5.1 Getting access to the silicon

Typical chip modules are fixed at the backside of the contact plate, covered with epoxy resin and then glued into the smartcard. So for depackaging, you first heat the card to soften the glue or use a knife to cut out the chip module. This method works for smartcards, small GSM chipcards and most pay-tv keys.

Then you have to get rid of the epoxy: This can be done with fuming nitric acid ($>$ 98 percent HNO_3). You can accelerate the process by heating the acid. To get usable results, you have to do several rounds of putting acid on the epoxy and washing away the dissolved material with acetone. The silicon will not get damaged by this, but you have to take care of the thin bonding wires connected to it. Even though there are machines for depackaging chips, chip analysis laboratories often prefer the manual method for small batches.

The chip surface will still be covered with a thin passivation layer that protects the silicon from the environment. That layer is consisting of silicon nitride or oxide. It is hard for amateur hackers to get rid of it, professionals may use dry etching with hydrogen flouride or use laser cutters to cut holes into the passivation, but those cutters are around $ 100.000.

5.2 Eavesdropping and fault generation

Most amateur hackers use invasive attacks only to reach bus lines with microprobing needles. They eavesdrop signals and apply glitches directly to the chip. For that, it is sufficient to break holes into the passivation layer using ultrasonic vibration or simply scratching. The possible attacks are similar to the non-invasive ones, but there is much more flexibility if you can attach directly to internal bus lines.

Microprobing needles are elastic metal hairs with a very thin tip (below 10 micrometers), and they are attached to high resolution positioners. You need a special microscope that leaves enough room between the lens and the chip to control positioning. By using second hand parts and designing your own micropositioners, you can assemble a microprobing workstation for less than $ 10.000, but professional versions cost up to $ 100.000.

There are far more possibilities for high level attackers with high end tools, but it is possible for amateur hackers to rent a lab with those tools for some time at chip testing laboratories.

5.3 *Advanced eavesdropping techniques*

For eavesdropping of many signals at the same time it is possible to "roentgenize" the chip using infrared laser light, which can reveal the logic states of single transistors. For faster signals, a standard method is to put crystals of lithium niobate on the chip. The optical properties of the crystals change with the voltage in their direct neighbourhood. For 5V TTL signals this works for up to 25 MHz signals.

To simplify read-out of interesting data, it is possible to reduce the clock frequency or do averaging over several measurements using repeated protocol replay. So a smartcard should block these conditions. Unluckily current non-volatile memory does not allow enough write cycles during lifetime for the complex logging that would be needed to block replay attacks. Ferroelectric RAM is an emerging technology that is likely to fix this problem. Another way to make replay attacks useless is to switch keys on a regular basis, so a spotted key will not be usable for long time.

Once the chip is depackaged, you can apply an electromagnetic field to part of the surface in order to modify the operation of certain gates. EPROM cells may be reprogrammed using focused UV beams.

Another effective but expensive tool is an electron beam tester (EBT). They work similar to scanning electron microscopes, but can show the voltage distribution on the chip. For a single point, this works up to several gigahertz, but only for periodic signals using stroboscopic sampling. For normal real-time sampling, only a bandwith of several megahertz is available. So if the device is protected against low clock frequency and replay attacks, eavesdropping using an EBT will be harder.

A robust low-frequency sensor is proposed in [KK99]: An external reset or power-up reset activates divisors on the clock signal, and only the reset triggered by the low-frequency sensor activates the rest of the chip. So if you destroy the sensor, the chip will no longer boot properly.

5.4 *Reverse engineering*

It is possible to do semiautomatic reverse engineering using a camera attached to a microscope and feeding the images into automated image recognition. The software may reconstruct the circuit down to the single transistor level, but it is much easier to spot standard cells or even standard processor designs because then it is not necessary to investigate the smaller details.

Using a special confocal microscope, the height information gets transformed into color information, so there are no blurred areas that are out of focus. It is possible for a funded attacker to remove the metal layers one by one to get complete insight on the structure. Some attackers can be confused by using non-standard cells or by polishing the top layer, so the layers below will not be visible through the height variation they cause.

Funded attacker can afford advanced techniques to make the implant mask and transistor threshold voltages visible (using so called dopant-selective crystallographic etch) or remove the top layers using hydrogen fluouride wet etching. Both techniques are used to make ROM contents visible.

5.5 *Modifying circuits with a FIB workstation*

A very convenient thing to have for a smartcard hacker is a focused ion beam (FIB) workstation. These are extremely expensive, but time on them can be rented at chip laboratories. Some are even found at universities. The FIB workstation uses an ion beam similar to the electron beam used in scanning electron microscopes. Therefore you can detect features down to 5 nanometers in size.

By increasing beam current, you can "drill" deep and narrow holes at any angle into the silicon. By injecting special gases it is even possible to deposit platinum or insulators on the chip surface. Navigation with 150 nanometer precision can be done using laser interferometers. It is now also possible to polish the back side of the chip to far below

0.1mm, so rear access is possible.

Most currently available smartcard processors have feature sizes of one or at least one half micrometer and only two metal layers, but future chips may require the use of even more expensive tools than FIB workstations (those are about $ 500.000 now). Most amateur hackers cannot afford to rent a FIB workstation for anything more than preparing chips for microprobing. An attacker who has constant access to a FIB workstation is able to virtually edit the circuits of the target chip, given enough time.

The only way to prevent FIB attacks is to cover the chip with something that cannot be removed without damaging the chip. But "conformeal glues" are still classified technology. If the card has its own power supply, it is much easier to use active intrusion detection.

A good way to keep low budget hackers out is to cover the chip surface with a sensor mesh where every other wire is connected to power and ground in turn (or transporting some non-secret signal). If the mesh gets damaged, the chip will no longer function properly. If a continuous power supply is available, the mesh can be used for active intrusion detection, eg. to trigger deletion of key data or a processor reset. This will not protect from FIB attacks, FIB workstations make it possible to drill fine holes between the sensor lines without damaging them.

A common FIB based eavesdropping attack is to destroy the instruction decoder and tap the bus while the instruction pointer passes every single byte of RAM. A way to prevent this is to use a short instruction pointer of only a few bits and another register as an offset. The idea is the same as in section 4.2.2.

Using FIB editing to reactivate a test circuit that was only switched off gives easy access to your data if you fail to do proper destruction of test circuits. A good way to destroy parts of the chip after quality control is placing them into the area where the die will be splitted into chips - or put the test circuits for each chip on the neighbouring chip.

6 *Conclusion:*

It is possible to make tampering harder without increasing the cost of the devices too much: Avoiding single points of failure, using segmented pointers and sensor meshes on the chip surface are examples. Other examples are hard to remove coatings (attempting to remove them should damage the chip), robust internal clock generation with randomized execution delays and current regulators right on the chip.

Fuses hidden in the lower layers are hard to analyze for reverse engineering, and the implant mask is harder to make visible than the arrangement of connection lines on the chip.

But you have to be aware that it is not possible to protect a smartcard for a long time from being hacked by a determined and funded attacker. So your system should cope with some hacked devices.

7 *References:*

[AK96] Ross J. Anderson, Markus G. Kuhn:
Tamper Resistance - a Cautionary Note,
The 2nd USENIX Workshop on Electronic Commerce Proceedings, Oakland, California, Nov. 18-21, 1996, pp 1-11, ISBN 1-880446-83-9.
http://www.cl.cam.ac.uk/Research/Security/tamper/

[KK99] Oliver Kömmerling, Markus G. Kuhn:
Design Principles for Tamper-Resistant Smartcard Processors,
USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA, May 10-11, 1999.
http://www.cl.cam.ac.uk/Research/Security/tamper/

LATEX