Advances in Smartcard Security

Marc Witteman

11

Introduction

Over the last decade smartcards have entered our global community. Although initially they were only used as simple phone cards they now support a large number of applications. Not all of them are successful yet, but their usage is still growing. Today over one billion smartcards are used in telecommunication (GSM), banking services and various other areas.

The original smartcard (or chip card) was nothing more than a memory chip that could hold a stored value. The only security feature was a simple mechanism that prevented cards from being filled up again after use. Even though that seemed to be sufficient for phone card applications it did not take talkative hackers long to break the mechanism.

Today's applications require more functionality and security and are much more complex. Therefore smartcards are now equipped with microprocessors and a significant number of security measures.

This article gives a technical overview of the advances and threats in smartcard security. First the designs of smartcard hard- and software are reviewed. Then the main attack classes and their countermeasures are discussed. Finally some suggestions are presented concerning how to achieve and maintain appropriate security in smartcard systems.

Smartcard Design

Hardware architecture

Smartcards come in different shapes. First of all there is a distinction between contact cards and contact-less cards. The first kind is easily recognised by the characteristic contact stamp that appears on both credit-card sized and SIM card sized versions. The second one is more difficult to identify because the chip may be hidden not only inside a credit-card sized container, but also in badges, car keys or labels.

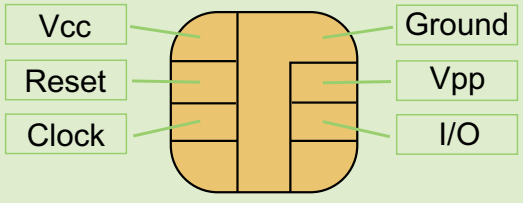

Figure 1 - **Contact Card Connections**

Secure tokens like dongles (hardware keys) and other types of smart tokens are related to smartcards, but mostly less complicated. They often implement proprietary hard-encoded encryption, although some newer versions use common algorithms like DES.

The connections used on contact smartcards are shown in Figure 1.

The Vcc and Ground contacts provide the power for the chip. The Reset contact facilitates a hard restart of all processes. The clock contact provides an external clock signal to the chip that serves as the heartbeat for the internal processes. The Vpp contact originally provided a higher EEPROM programming voltage, but is no longer in use. Finally the I/O contact is a serial channel for bi-directional communication between smartcards and terminals.

Contact-less cards use electromagnetic induction to provide power from terminal to smartcard as well as for bi-directional data exchange.

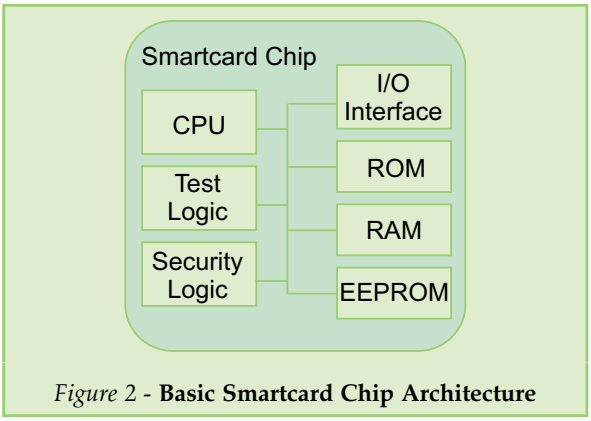

Figure 2 - **Basic Smartcard Chip Architecture**

Whatever the container is, the functionality of the smartcard is embodied in a single small chip. Figure 2 shows a basic schematic of the chip.

Unlike many other chips, a smartcard chip is a complete computer in itself. It combines functions that are often distributed over separate components:

- **CPU (Central Processing Unit)**: the heart of the chip, all computational work and data exchange goes via this function. Sometimes a cryptographic coprocessor is included. Most smartcard CPUs run on a clock frequency of 3.57 MHz.

- **Test Logic**: a verification function only used during the production process to test all internal circuits for manufacturing faults.
- **Security Logic**: a continuous function that checks environmental conditions that could jeopardise the security of the smartcard.
- **I/O Interface**: a communication function that takes care of receiving external commands and sending back responses using a serial communication protocol.
- **ROM**: the permanent memory of the chip. It can contain parts of the operating system and self test procedures. The memory size is typically 32 Kb.
- **RAM**: the CPU's scratch pad memory. This is used for storing temporary or intermediate data like session keys, internal variables and stack data. The memory size is typically 1 Kb.
- **EEPROM**: non-volatile updateable memory. It is used for storing application data like keys, PINs, balances, phone numbers and sometimes application or even operating system code.
- **Data Bus**: the transfer channel within the chip. All information exchanged between the various functions passes through this channel.

All these functions have to be implemented on the small surface available for smartcard chips (typically not larger than 25 mm^2). Smartcard chips do follow the tendency towards miniaturisation that is common practice in the chip industry though. State-of-the-art smartcard chips use feature sizes less than 0.2μm and up to 7 stacked layers of metal and silicon.

Software Architecture

Early processor cards used a monolithic software model: operating system and application functions were closely interwoven. Nowadays most smartcards use a more modular software design and also application separation. A popular smartcard operating system is called *Java Card* and uses proven security concepts from the Java language. This operating system allows for flexible application design. Applications can be developed and loaded after card manufacturing or even post-issuance.

Modern smartcard operating systems use life-cycle management. This process restricts the actions that can be performed in the smartcard. The system must for instance be quite open during the manufacturing process to facilitate configuration, but much more closed during field operation to avoid fraud.

Data stored within a smartcard is organised in a nested file system. The EEPROM is used similarly to a hard disk and can contain files and directories with user and application data.

The cryptographic functions may be implemented partly in hardware, but there is always a software program that controls the execution and that is stored either in the permanent or non-volatile memory.

Terminals (or card readers) talk to smartcards by means of commands. Figure 3 shows the command structure.

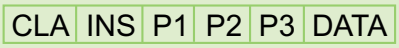

Figure 3 - **Command Structure**

Commands have a five-byte header followed by an optional variable-length data part. The first two bytes CLA and INS specify the *Class* (context) and *Instruction*. This structure allows the definition of 216 different commands. The next two bytes P1 and P2 are *Command Parameters*. P3 indicates either the *Length* of the data included in the command, or the length of the requested response data.

Some common commands are:

- **SELECT**: open a file or directory.
- **READ**: read a file
- **UPDATE**: change the contents of a file
- **AUTHENTICATE**: authenticate the smartcard to the external world
- **VERIFY**: check a cardholder's PIN code.

Smartcards enforce access conditions to protect the data content of files. Users (terminals) can only access files for reading or writing if the proper access conditions are fulfilled. Access conditions may require PIN verification or external authentication. More information on smart card design basics can be found in [1].

Smartcard security threats

Smartcards are popular targets for attackers, for various reasons:

- Successful attacks enable fraud and are valuable; professional attackers can make a business case.
- Smartcards are cheap and easy to obtain; attackers can easily acquire some samples for training.
- Smartcards are portable; the attacker can easily bring them to a hostile environment and control the conditions.

Manufacturers are well aware of these temptations and pay particular attention to secure their products. In practice 100% security is never possible and designers and attackers of secure systems continuously challenge each other and new advances are made.

We can distinguish between three basic attack classes:

- **Logical Attacks**: exploits that use bugs in the software implementation.
- **Physical Attacks**: exploits that use analysis or modification of the smartcard hardware.
- **Side Channel Attacks**: exploits that use physical phenomena to analyse or modify the smartcard behaviour.

The following sections discuss these attack classes and counter measures taken by the industry.

Logical Attacks

Smartcards have a single communication channel to exchange data with a terminal (smartcard reader). This channel is a serial interface where commands can be issued that the smartcard must perform. Although smartcards are small computers, there is an amazing number of command options that may be supported. Due to this complexity and time-to-market constraints it happens that hidden flaws that do not affect the normal behaviour, remain undetected during security tests. Logical attacks abuse these flaws to trick the smartcard into surrendering confidential data or allowing undesired data modifications.

Logical Security Threats

Potential logical flaws may relate to many different aspects.

Hidden Commands

Smartcard operating systems can technically distinguish between more than 65000 commands. Although practical use may require only a few commands there may be some remaining and active commands from an initialisation phase or from a previous application. These commands may be abused to retrieve data from or modify data in the smartcard.

Parameter Poisoning and Buffer Overflow

Commands are accompanied by a number of parameters that specify the exact request. A disallowed parameter value or length may not be rejected, but misinterpreted and lead to surprising results. A simple, but sometimes effective, example is a file read command where offset and requested length exceeds the actual file size.

File Access

Smartcard file systems have detailed permissions on files and directories. For each command access permissions determine the security procedures to access a file. It may happen that the access permissions implemented allow more access than needed for specific files, thus creating a security hole. Complex interactions can occur when several distinct applications need to be accessed during one session, and operating systems may confuse access permissions.

Malicious Applets

Smartcards that support multiple applications need to ensure application separation. The operating system should create a virtual environment where applets cannot harm each other. If an attacker manages to download a rogue applet, or abuse a flaw in one applet, he may be able to compromise another security sensitive applet.

Communication Protocol

Information exchange between smartcard and terminal is dictated by a communication protocol that handles data flow control and error recovery. By sending messages outside the scope of the current state it may be possible to trick the smartcard into revealing secrets. A notorious example uses an error correction facility in the communication protocol. Smartcards use a small message buffer within the RAM memory to store operation results. They also keep a length field that indicates the length of the available buffer data. Whenever a message is not correctly received the receiver can request re-transmission of the message. If a smartcard reader were to ask for re-transmission of a message that had not yet been sent at all, a sloppy smartcard implementation may decide to send the buffer anyway. If at the same time the length field is not properly initialised the implementation may send a large part of, or even the entire remaining memory. Such a memory dump results in a complete surrender of all confidential data and secrets. Figure 4 shows how a message buffer is extended over the remaining memory due to an improper value of the length field (LEN).

Crypto-Protocol, Design and Implementation

Cryptographic protocols handle consecutive cryptographic operations to perform transactions. If such a protocol is not carefully designed it may present opportunities for attackers to per-

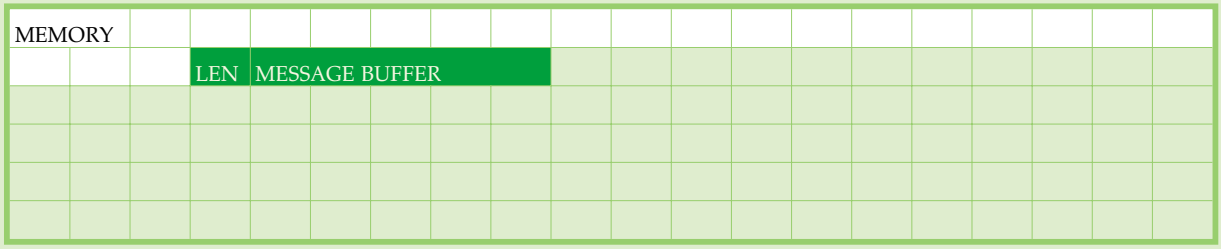

Figure 4 - **Memory Dump**

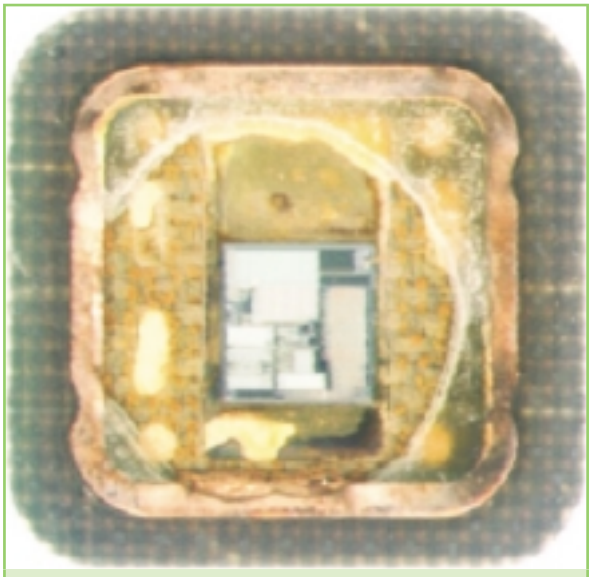

Figure 5 - **Etched Smartcard Chip**

form replay attacks or related exploits. Some cryptographic schemes have fallback methods to enhance reliability in case of technical problems. These fallback methods may be less secure and attacks may benefit from creating fictitious malfunctions. Furthermore, many cryptographic algorithms are still proprietary and have never been publicly reviewed. They may be flawed and eventually be broken when the design leaks out. Finally, number generators may not generate sufficient randomness, thus becoming predictable.

Note that some of these attacks are not exclusively smartcard attacks, they are applicable to other systems as well. An important aspect of all these attacks is the fact that they are rather cheap in terms of required equipment. An attacker needs only a smartcard, a card reader, an ordinary PC and maybe some manuals or standards. Another attractive aspect is the non-destructive nature of these attacks and the ease of reproduction. Fortunately, simple exploits in this category are rare and attackers will need a lot of persistence and imagination to find a backdoor.

Countermeasures for Logical Attacks

The sensitivity to logical attacks is very much dependant on the complexity of the software. Software developers know that the number of bugs grows with the size of the code. Some strategies to combat software bugs (including security flaws) are:

- **Structured Design**: create software in small functional building blocks that can more easily be understood and validated.
- **Formal Verification**: use mathematical models to prove the soundness of functions.
- **Testing**: perform experimental validation of the implementation.

In the field of smartcards there are a couple of trends that can be categorised likewise:

- **Standardisation of Interfaces and Applications**: re-use of proven software decreases the chance of flaws.
- **Convergence to the Java Card Operating System**: an object-oriented language that was designed for security is conceptually more secure than the older monolithic operating systems without application separation [2].
- **Popularity of Evaluation Labs:** a growing number of card manufacturers and card issuers use evaluation labs to get a (formal) report or certificate.

Despite these trends smartcards are far from immune to logical attacks. The growing software complexity will always bring the risk of introducing new flaws. Careful design and validation may reduce the number and increase the difficulty of exploiting the flaws, though. Unskilled attackers may then no longer be able to find exploits.

Physical Attacks

A smartcard chip may appear to be an electronic safe, but it is actually not all that secure. Although all its functions are encapsulated in one chip it is possible to reverse engineer them. Physical attack methods require high-end lab equipment, but do provide powerful tools to perpetrate successful exploits.

Physical security threats

Physical attacks can be performed by means of a variety of methods and tools [3].

Chemical Solvents, Etching and Staining Materials

Etching materials are able to decapsulate and accurately de-layer smartcards chips. Figure 5 is a photo of an etched smartcard chip. The chip surface reveals the various building blocks in the chip. After this process the chip is accessible for optical or electrical analysis. The epoxy that fixates the chip into the card can easily be dissolved, but the removal of metal and silicon layers requires quite aggressive and dangerous chemicals that should only be handled by experts in a chemistry lab. Because modern chips contain multiple layers this is an essential step in optical reverse engineering. Staining is an advanced etching technique that uses differences in etching speed to reveal subtle material differences that define the ones and zeroes in some ROM memories.

Microscopes

Optical as well as *Scanning Electron Microscopes* (SEM) can be used for optical analysis and reverse engineering. Although chip feature sizes are well below one micron they can still be seen with a good optical microscope and it may be possible to reverse engineer hardware scramblers, crypto-en-

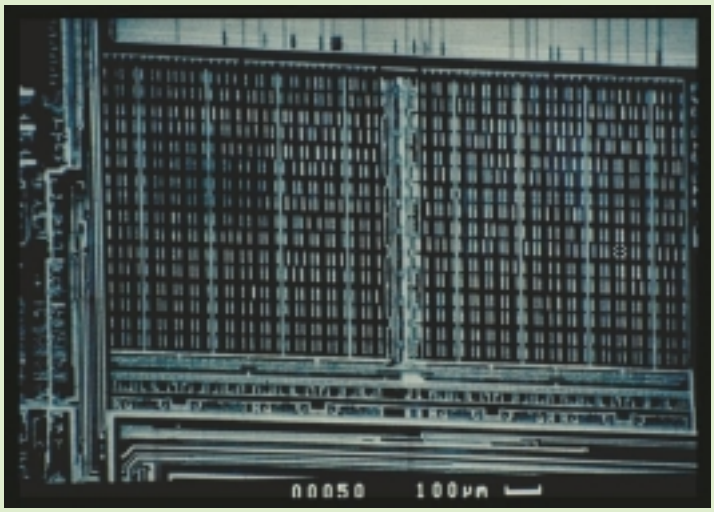

Figure 6 - **RAM Voltage Contrast**

gines or hard-wired ROM. Automated tools can reconstruct complete circuits, or operating system source code from the ones and zeroes in a ROM mask. SEMs can see much smaller details, but are also capable of visualising life-effects of a running circuit. *Voltage Contrast* is a SEM application that can see high and low power values on the chip wires. A carefully prepared chip that is still capable of performing its electronic functions can be analysed to reveal active sections in the chip and potentially even running code or passing data values. Figure 6 is a voltage contrast photo of an active RAM memory. The values of the memory cells can actually be seen.

Probe Stations

This type of equipment allows tiny probe needles to be positioned on arbitrary wires on a naked chip. Provided that a chip is still performing its electronic functions it is possible to create new channels to the outside world. If the data bus can be located, probe needles may be able to tap all data exchanges between the CPU and the memories. In combination with a logic analyser it is possible to retrieve full running program code and program data including keys. Vice versa it may also be possible to force a wire to accept data that effectively overrules the original data. In that manner micro-instructions can be changed so as to cause the processor to take a complete different execution path with all possible consequences.

Focused Ion beam (FIB)

This variation on a Scanning Electron Microscope shoots *ions* in stead of electrons and is not only able to make small details visible, but also to make changes to circuitry. By adding different gasses to the ion beam it is possible to deposit material that creates wires, insulators or even semiconductors. This way blown fuses of test circuits can be reconnected, or hidden internal signals can be forwarded to external wires. In multi-layer chips it may be possible to surface 'buried' wires by creating a sort of tunnel. Also wires that are too thin and fragile to put probe needles on can be strengthened and enlarged to form a *probe pad* by putting on extra material with the FIB. Figure 7 shows a cross-shaped probe pad made to ease probing a thin wire.

Although physical attacks are extremely powerful they also have a disadvantage: they are invasive and often destructive. As the attacker can often not re-use the device it is not always an attractive approach. Also, the high-end lab equipment and expertise required may be a problem. On the other hand an increasing number of commercial labs offer both equipment and service as these activities also serve chip failure analysis and chip repair.

Countermeasures Against Physical Attacks

Chip manufacturers have attained a significant improvement in physical security over the most recent years. This is a remarkable change. It is only a couple of years ago that retired machines from regular chip production were in general used to make smartcard chips. This was caused by the low chip prices and limited functionality needed for the chips. Today the smartcard market is a mass market and functional complexity has grown hugely. For that reason manufacturers can afford to use new advanced equipment and highly sophisticated chip designs.

Specific areas of improvement are:

- **Feature Size**: in 5 years' time the size of transistors and wires on the chip surface has shrunk from more than 1 μm to less than 200 nm. This size is too small for optical microscopes to analyse and too small for probe sta-

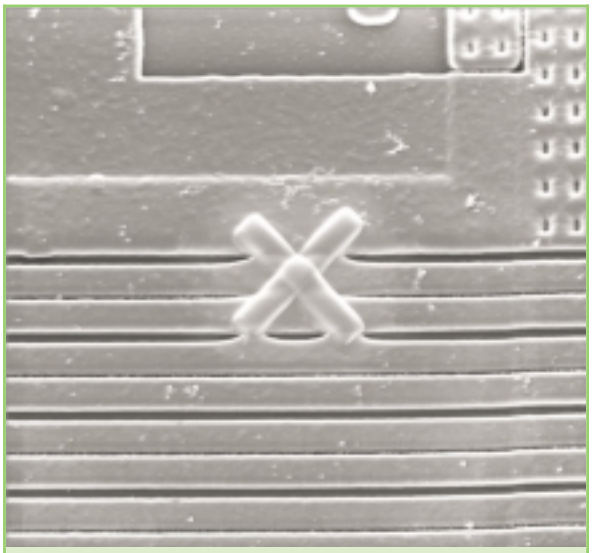

Figure 7 - **FIB Cross-Shaped Probe Pad**

tions to put needles on. Sophisticated microscopes and Focused Ion Beams can still handle this size, though.

- **Multi-Layering:** today's smartcard chips use multiple layers. Not only is the number of semiconductors that can be produced, larger, but is it also possible to hide sensitive data lines (buried layers) underneath other layers that contain less sensitive connections.
- **Protective Layer**: in order to prevent analysis of live data processing it is possible to use a top layer that contains an active grid carrying a protection signal. Interruption of that signal will cause the chip to erase its memories and halt. However, skilled attackers might still be able to make a bypass through the grid and then penetrate the protective layer. Therefore, advanced grids would use a large number of seemingly non-correlated and frequently changing signals. This will significantly reduce the attacker's ability to access underlying lines by means of FIB modifications.
- **Sensors**: signals that measure environment variables such as light, temperature, power supply and clock frequency can be used to disable the chip as soon as out-off-bound conditions are detected. This will reduce the attacker's possibility to do live data analysis on a prepared chip. On the other hand they may also affect the reliability of the chip and for that reason be tuned quite fault-tolerant.
- **Bus-Scrambling**: the data bus between various building blocks (e.g. processors and memories) can be scrambled using a sophisticated non-constant scrambling technique. An attacker attempting to interpret the bus data needs to do a full reverse engineering of the scrambler logic.
- **Glue Logic**: instead of placing functional blocks in separate sections on the chip it is also possible to mix it all up and create glue logic. This way an attacker will no longer be able to easily identify the functional building blocks by analysing the physical structures on the chip.

Altogether there are many ways to reduce the possibilities of physical attacks to succeed. Nevertheless not all manufacturers use all of these options, or use them only for their most advanced and expensive devices. Many smartcard chips existing on the market today do not yet benefit from the newest technological advances. It is noteworthy that analysis techniques are improving as well and becoming more accessible.

Side Channel Attacks

Despite the complex chip designs we have to realise that integrated circuits are just a whole bunch of switching semiconductors. These semiconductors are sensitive to basic physical phenomena like electric power and radiation. Although the chips are designed to process programmed stimuli and communicate only via restricted channels, they are in fact quite sensitive to variations in their environment. They also produce signals apart from those that were intended.

Side Channel security threats

Side Channel Attacks can be subdivided into *Side Channel Analysis* and *Side Channel Manipulation*.

Some physical phenomena that can be used for Side Channel Analysis by observation of the behaviour of an electronic circuit are:

- **Power Consumption**: semiconductors use electric current during operation. The total amount of power consumed by a chip is very much dependent on the ongoing process. Measurement of the power consumption can reveal detailed information about the information being processed.
- **Electromagnetic Radiation**: Every switching transistor produces a bit of electromagnetic radiation. Just like power consumption this information can in theory provide a complete picture of the ongoing processes.
- **Time**: Microprocessors need time to complete their tasks. The amount of time may be variable and related to process parameters.

Physical phenomena that can aid Side Channel Manipulation by disturbing electronic circuits are:

- **Voltage**: most electronic circuits are designed to operate from a defined and constant supply voltage. Sudden changes to the power supply (power glitches) may change the behaviour of the chip and trigger alternative behaviour.
- **Electromagnetic Radiation**: A strong electromagnetic pulse can induce signals into the chip wires that may damage the chip, but also change its behaviour.
- **Temperature**: electronic devices have a limited temperature range for operation. Outside the boundaries it may be possible to change their behaviour.
- **Light and X-Rays**: semiconductors are sensitive to light. A suitably directed beam of light will affect a region of a chip, possibly resulting in behaviour changes.
- **Frequency**: microprocessors are designed to operate within a nominal clock frequency range. Above the maximum frequency switching errors may occur in complex instructions that need a bit more time. A very low frequency (or even single stepping) may also offer interesting observations to an attacker.

Side Channel Attacks are attacks that use these physical phenomena to analyse or manipulate the behaviour of a smartcard chip. Although

they are related to the physical attacks that were discussed before, they are essentially different in operation because they are non-invasive. Side Channel Attacks can be practised without physically opening the device and without damaging it.

Although there is a rich spectrum of phenomena to use to retrieve information or manipulate chip behaviour, the problem is to apply these attacks successfully. The analysis techniques need to analyse huge amounts of data in order to filter out the essential information. The manipulation techniques need to tune the magnitude of disturbance and focus their timing within the processes to attack.

Over the past five years two types of side channel attacks have been developed and applied with great success:

- Differential Power Analysis
- Power glitching

These two attack types have had a lot of impact on the smartcard industry because they were relatively cheap to perform and offered a high chance of success. The following subsections discuss these attacks in more detail.

Differential Power Analysis

It has been known for quite a while that it is possible to measure the power consumption of an electronic device (by means of an oscilloscope) and observe its behaviour. Figure 8 shows a power trace of a device that performs an encryption using the DES algorithm.

It is clear that the power consumption is not constant and reveals some patterns. Knowing that DES takes 16 rounds to encrypt the input data it is possible to identify the rounds in the 16 repeating patterns in the power trace. Although this is an interesting observation it does not give an answer to the more important question: what is the key used for this encryption?

A few years ago Paul Kocher published the concepts of *Differential Power Analysis* (DPA) [4]. This technique can retrieve the key of a cryptographic algorithm by analysing a number of measured power traces. An attacker only needs to know either the clear text (input) or cipher text (output) of the algorithm. The basic idea behind the attack is the assumption that there is a correlation between data values being processed by the device and the power consumption. In other words: it is assumed that processing a bit value zero uses less energy than processing a bit value one (or vice versa). Figure 9 shows two power traces taken from an algorithm running twice on different input values.

The power trace shows the measured power consumption during eight clock cycles. Individual clock cycles can often easily be recognised due to the peaks caused by a large number of transistors that all start switching at the beginning of a new cycle. Although the two traces are almost identical it appears that there is a small difference in the third cycle. Figure 10 shows the difference between the two traces.

The explanation for the peak in the differential trace is given by the different input data being processed. In fact the input data to the algorithm is being processed exactly at the third cycle in these traces. The distinctive input values have resulted in a small difference in power consumption and can be identified by inspection of differential traces. It appears that by computing differential traces it is possible to identify the clock cycles where input data is being processed. Moreover, by considering all input bits to a cryptographic algorithm and creation of differential traces for each pair (a trace for a bit value zero and a trace for a bit value one), it is possible to identify the exact timing of their appearance in the program code. In situations where noise prevents the recognition of peaks in the differential trace it is possible to increase the number of samples and compute a differential trace out of many individual traces in stead of just two.

An important observation now is that the internal processing of an encryption algorithm can be studied. This is a dangerous conclusion carrying implications for security requirements. Cryptographic algorithms must use a sufficient key length (many possible key values) to ensure that an exhaustive key search is not feasible. Cryptographic implementations break the complete algorithm up in many small steps that can be performed by processors. These small steps tend not to use the complete key, but just a part of it. DPA allows observation of the outputs of these small steps followed by an exhaustive search on the small number of key values.

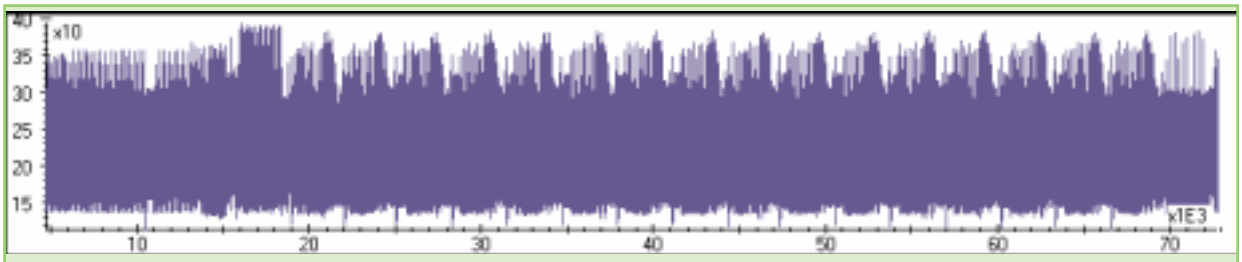

Figure 8 - **Power Trace During a DES Operation**

For as long as no processors are able to perform a full encryption in a single atomic step it seems that all algorithms must be susceptible to DPA [5]. Although the development of such attacks is quite complicated, the application of the attacks is easy and needs only small investments. The required equipment is limited to a PC and medium quality oscilloscope. For that reason solving the DPA issue became one of the highest priorities for smartcard manufacturers.

Countermeasures for DPA and other types of Side Channel Analysis Attacks

The smartcard industry has developed quite a few more or less successful defences against Power Analysis and other types of Side Channel Analysis attacks. Basically there are three levels of defence: hardware, software and application level.

Typical hardware countermeasures are:

- Lowering the power signal by balancing circuits and reducing electromagnetic emissions (by means of metal shields).
- Increasing amplitude noise level by performing concurrent random processes. The circuit used for the internal generation of a programming voltage could for instance be used as a concurrent noise generator.
- Introducing timing noise with process interrupts and variable clock speeds. It is essential for the computation of differential traces that traces can be aligned; the occurrence of processor steps should be synchronised before adding the traces. Timing noise will prevent or at least hamper a good alignment of traces.

Good hardware countermeasures have the advantage that the smartcard susceptibility to side channel analysis is less dependent on changes in the software. A disadvantage is the fact that hardware countermeasures can only *reduce* the susceptibility to side channel analysis, but not eliminate it completely. They just make the attacks more difficult by reducing the signal to noise ratio.

Software countermeasures include:

- Reducing relevant signals by random process ordering. Parallel substitutions in an algorithm (like the S-boxes in the DES) can for instance be performed in a random order. The number of substitutions being reordered then divides the signal produced by one substitution.
- Adding timing noise by random delays or alternating paths. The timing noise will hamper the alignment of traces, and deteriorate the quality of the differential trace.
- Eliminating time dependencies in key material and intermediate values. Simple power analysis by visual inspection of traces is possible when the duration of functions is depending on key values. A time-constant implementation of key operations will prevent this easy attack.
- Blinding intermediate values with random values. Power leakage relates to data hamming weight (number of bit values set to one). If random data are added to the actual data and subtracted later then the intermediate path will not leak useful information. Unfortunately this blinding will cause non-linear intermediate functions to produce wrong results. These functions must therefore carefully be designed to compensate for the deviations caused by the random data.

Some of these measures just aim to decrease the signal to noise ratio (like the hardware countermeasures), but some are more fundamental. In principle it is possible to implement 'perfect' software countermeasures. They would completely eliminate the emission of useful information from the side channel, and would even be successful on extremely leaky hardware platforms. Unfortunately such countermeasures are expensive and hard to maintain because they are

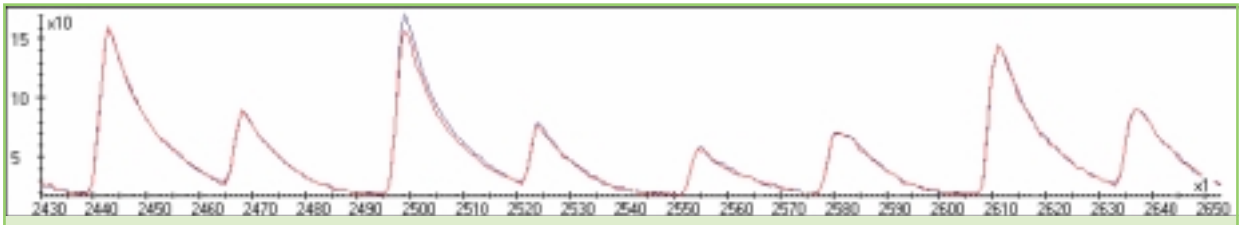

Figure 9 - **Two Overlapping Power Traces for Different Input Values**

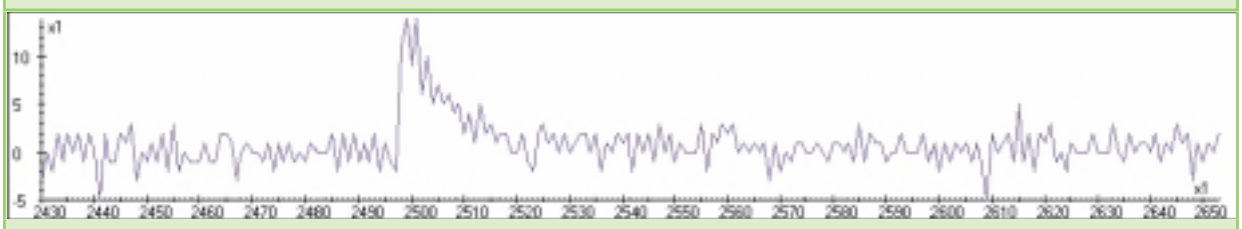

Figure 10 - **Differential Power Trace for Different Input Values**

tailored to the chosen algorithm and quite difficult to design.

Application level countermeasures include:

- Retry-counters that limit the number of samples that an attacker can take. A PIN verification that blocks after three successive failures is a useful protection against differential analysis.
- Limited control and visibility of input and output of cryptographic algorithms. An attacker may not be able to perform a differential analysis if only part of the input can be chosen, or only part of the cryptographic result is returned.

These are generally simple countermeasures that basically deny the requirements for side channel analysis. The disadvantage is a negative reliability impact and the need to change existing protocols.

Generally speaking the side channel analysis problem is being tackled, but not solved. The industry still has a tremendous job to do to reach adequate and lasting solutions.

Power Glitching

Microprocessors are designed to operate from a stable voltage. Interruptions of the power supply are likely to crash running applications or reset the circuit. However, a short and well-tuned glitch may introduce a single computational fault while the processor continues to execute the running program code. Consider for example the CPU reading the contents of a memory cell. A transistor measures the value stored in the cell by using a threshold value to decide whether it reads a logical zero or one. A sudden power glitch will affect both the stored and the threshold values. Different internal capacities will cause the values to be influenced differently, possibly resulting in a misinterpretation of the actual value. Figure 11 illustrates that a low voltage corresponding to a logical zero may be below the threshold at normal operating conditions, but above threshold during the short power dip.

Many cryptographic algorithms are susceptible to this type of fault injection. A technique called *Differential Fault Analysis* (DFA) is used to compare correct encryptions to faulty encryptions

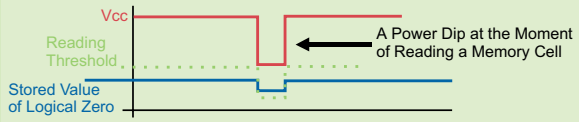

Figure 11 - **Power Glitch During Memory Reading**

and extraction of the secret key from the comparison. Some algorithms can only be attacked if a precise intermediate value in the process is hit, while others are less critical and can be hit about anywhere in the process. Generally DFA requires the possibility of encrypting the same cleartext twice, resulting in one correct and one faulty ciphertext.

A second application of fault injection occurs around crucial decisions in security processes. If an application performs a security check like a PIN verification it might be attractive to hit the device at the very moment it decides to continue or abort the process. An attacker may be able to trick the processor into interpreting a PIN verification failure as a success. An even more rigid variation would be to completely turn off the power as soon as the processor is about to write a failure into memory, thus preventing a retry counter to overflow.

A third application of glitching aims at manipulating the communication function. The communication protocol is designed to read a few bytes from the smartcard memory and send them to a terminal. But, if the fault injection succeeds in hitting the send limit counter it might result in dumping the complete memory to the serial interface.

All these attack variations have been applied successfully to smartcards.

Countermeasures for Power Glitching and Other Types of Side Channel Manipulation

Power Glitching and other Side Channel Manipulation techniques all try to change environmental conditions. The common strategy against these attacks is the rigid use of sensors for voltage, frequency and temperature. Unfortunately a strict sensor setting affects the reliability as well, potentially causing malfunctions in some terminals or climates. Furthermore, it is not possible for sensors to detect all induced signals. Electronic circuits will never be fully immune to sig-

Attack Class	Equipment	Cost	Success Rate	Development Time	Execution Time
Logical	PC	$1-10K	Low	Weeks	Minutes
Physical	PC, Probe Station, SEM, FIB, Microscope, Chemistry Lab, etc	$100K-1M	High	Months	Days
Side Channel	PC, Oscilloscope, Function Generator	$10-100K	Medium	Months	Hours

Table 1 - **Attack Statistics**

nals that are injected by means of induction, or carefully tuned power glitches. It is important to implement software and application countermeasures to detect and recover from fault injection.

With respect to software countermeasures it is possible to carry out fault detection by checking crucial program flow decisions and cryptographic results. Computing the results twice and comparing both results is a way of checking the validity of the results. A potential problem is the possibility that the same fault was injected twice and could not be detected. It is better (if possible) to reverse compute the input from the result and compare that to the original input. A reverse computation is often different and more difficult to manipulate in the opposite direction. It is important to consider carefully which parts of the code are crucial from a security perspective.

Despite the growing awareness in the smartcard industry about potential solutions for Side Channel Manipulation many smartcards are not immune to this class of attacks today.

Attaining Security

Designers of secure applications use smartcards despite the many vulnerabilities. Alternative solutions have there own security pitfalls, or are even inherently less secure. This section provides some practical tips for application designers to attain an adequate level of security.

The Attacker's Business Case

A designer should always consider the business case of the attacker. Although fun and fame may be driving factors for some, most serious threats come from attackers that seek financial benefits. These attackers will consider the balance between their costs and revenue.

Table 1 gives a generalised overview of statistics for these attacks.

Although Logical attacks seem to be more interesting in terms of investment the chances for success are often slimmer. Also the more expensive equipment need not always be bought, but can quite often be leased instead.

Basically, system designers can prevent most attacks by reducing their financial attractiveness. Technical countermeasures contribute to that goal by increasing the effort and cost needed to mount successful attacks. More important though, is the tuning of business models such that the gain obtained through successful attacks is limited.

Security cost

Development of secure smartcard solutions carries a price. Although security is an inherent aspect of hardware, software and applications, there are differences between various players. The manufacturers of hardware and software are mainly situated in Europe and Asia (only few companies in smartcard manufacturing are situated in North America). The European manufacturers are generally more advanced in security features whereas the Asian companies most often offer cheaper products.

In terms of security features and development costs there seems to be a relation between the cost of the attack and the cost of counter measures: counteracting physical attacks is often more expensive than counteracting logical attacks. This is related to the fact that both designer and attacker use similar equipment and methods.

Although manufacturers may have to invest heavily in countering security problems, this does not necessarily lead to large price increases for their products. Marketing and volume are more important aspects in setting a price. E.g., GSM SIMs that include some defences against DPA often cost just a 10% more than less secure equivalents.

Design Steps

A smartcard application designer uses off-the-shelf smartcard products and designs system, software and protocols to implement the application. Despite the many threats he should still deliver a sufficiently secure system. Here are some steps he could take to reach that target:

- Define the level of security needed for the application and specify security requirements. Also consider technical, commercial and public relations (brand value) costs of potential security problems.
- Perform a risk analysis and assess the security threats.
- Analyse the business case of the attacker. Consider various types of attackers ranging from gentle hackers to criminal organisations. Also understand that fraud often involves personnel within the company.
- Select smartcard solutions that deliver the level of security required.
- Include detection mechanisms that find security breaches. Fraud management systems combine various information sources to reveal security problems before they become business problems.
- Design recovery and fallback solutions. Procedures should be in place to repair or replace system parts that suffer from security problems.
- Consider overall security; smartcards are just one link in a complete transaction chain. Make sure other parts are equally secure, or compensate for known weaknesses.

Conclusion

The smartcard market has experienced a spectacular growth over the past few years. Smartcards have evolved from simple devices made with inferior equipment into complex high tech security solutions. With respect to end-to-end security no other security solutions nearly as good and affordable as smartcards exist.

Along with their growing popularity there has been a corresponding growth of interest in their security weaknesses. This article has outlined the potential and realistic smartcard attacks. Three attack classes were considered: Logical, Physical and Side Channel Attacks. Each class has its own characteristics and dangers. Even though smartcards are specifically designed for security there is no such thing as perfect security. Every new security design will eventually face its threats.

There is no easy recipe to counteract smartcard security threats, but it is possible to minimise the risks. Organisations implementing smartcard solutions must realise that information security is an ongoing process that requires more than just a few actions.

At the design stage the system architects should consider the latest threats and state-of-the-art security solutions. Systems depending on smartcard security should incorporate mechanisms for detection of and recovery from security problems. An important step prior to service introduction is a security evaluation where experienced labs test the product security against the newest attacks and validate the effectiveness of security measures. Regular studies to emerging threats, possibly enhanced with repetitive evaluations, should be used to ensure that the stipulated security is maintained at a desired level throughout the system life cycle.

In spite of all the threats and attacks it is possible to achieve practical smartcard security, but this pleasure is neither easy nor without cost.

References

[1] W. Rankl, *Smart Card Handbook*, Wiley & Sons, ISBN 0471988758, 2nd edition, 2000

[2] Zhiqun Chen, *Java Card for Smart Cards: Architecture and Programmer's Guide*, Addison-Wesley, ISBN 0201703297, 2000

[3] Ross Anderson & Markus Kuhn, *Tamper Resistance - a Cautionary Note*, in proceedings of the Second USENIX Workshop on Electronic Commerce Proceedings, Oakland, California, 1996, ISBN 1-880446-83-9.

[4] Paul Kocher, Joshua Jaffe and Benjamin Jun, *Differential Power Analysis*, in proceedings of Advances in Cryptology, CRYPTO '99, Springer Verlag, 1999.

[5] Marc Witteman, *Power Analysis on Unknown Crypto Algorithms*, presented at the workshop Cryptographic Security Aspects of Smart Cards & the Internet, Amsterdam, June 2002.

Marc Witteman was awarded his MSc degree in Electrical Engineering from the Delft University of Technology in the Netherlands. In 1989 he joined KPN where he initially worked on GSM development. Later he worked in the field of testing theory, which resulted in a couple of scientific papers and patent applications. In 1994 he moved to the area of smartcards. During 1996 and 1997 he was affiliated with the ETSI standardisation body where he headed a smart card team. In 1997 he moved to TNO where he became a research leader in smart card security projects. He developed and practised many new smart card evaluation methods. In 2001 he started his own security consulting company called Riscure.

He can be reached at witteman@riscure.com.