Known Attacks Against Smartcards

White Paper

Discretix Technologies Ltd.

Author: Hagai Bar-El
Title: Information Security Analyst
Email: hagai.bar-el@discretix.com
Tel: +972-9-8858810
www.discretix.com

ABOUT THIS DOCUMENT

This document analyzes, from a technical point of view, currently known attacks against smart card implementations.

The purpose of this analysis is to give the necessary background for the assessment of the mechanisms that can enhance the security of smart cards.

This document is mainly intended for people who are considering the use of cryptographic modules and who need to compare several options with respect to their security.

REFERENCES

1. *A method for resynchronizing a random clock on smart cards*, By Didier Moyart and Régis Bevan, Oberthur Card Systems
2. *A Timing Attack on RC5*, By Helena Handschuh and Howard M. Heys
3. An Overview of Smart Card Security, By Siu-cheung Charles Chan
4. Biham-Shamir Differential Fault Analysis of DES, By Eli Biham and Adi Shamir
5. Defending against DFA, By Sandy Harris
6. Design Principles for Tamper-Resistant Smartcard Processors, By Oliver Kommerling and Markus G. Kuhn
7. *Differential Power Analysis*, By Paul Kocher, Joshua Jaffe, and Benjamin Jun
8. *DPA Q&A*, By Paul Kocher, Joshua Jaffe, and Benjamin Jun
9. Hacking EPROM based microcontrollers in general, Author unknown
10. Hardware Security of Advanced Smart Card ICs, By Koninklijke Philips Electronics
11. Improving Smartcard Security Using Self-timed Circuit Technology, By Simon Moore, Ross Anderson, and Markus Kuhn
12. Introduction to Differential Power Analysis and Related Attacks, By Paul Kocher, Joshua Jaffe, and Benjamin Jun
13. *Java Smart Cards*, By Y. L. Chan and H. Y. Chan
14. *Low Cost Attacks on Tamper Resistant Devices*, By Ross Anderson and Markus Kuhn
15. *New Attacks to Public Key Cryptosystems on Tamperproof Devices*, By Feng Bao, Robert Deng, Yongfei Han, Albert Jeng, Desai Narasimhalu and Teow Hin Nagir
16. Probing Attacks on Tamper-Resistant Devices, By Helena Handschuh, Pascal Paillier and Jacques Stern
17. Security Policy For Schlumberger Sema Cryptoflex 8K Smart Card, By Schlumberger
18. Smart Card Technology and Security, Author unknown
19. Smart Cards and Private Currencies, By J. Orlin Grabbe
20. Four-way encoding aims to beat smartcard hackers, Author unknown
21. Tamper Resistance - a Cautionary Note, By Ross Anderson and Markus Kuhn
22. *Tamper Resistance - A Second Opinion*, By Semiconductor Insights Inc.
23. Tamperproofing of Chip Card, By Ross Anderson
24. Which Smart Card technologies will you need to ride the Information Highway safely, By Patrice Peyret, Head of Research & Development, Gemplus
25. Breaking Public Key Cryptosystems on Tamper Resistant Devices in the presence of Transient Faults, By F. Bao, R. H. Deng, Y. Han, A. Jeng, A. D. Narasimhalu and T. Ngair
26. Cryptanalysis of the m-Permutation Protection Schemes, By Hongjun Wu, Feng Bao, Dingfeng Ye and Robert H. Deng
27. *Introduction to Side-Channel Attacks*, By Hagai Bar-El, Research Group, Discretix Technologies.

KNOWN ATTACKS

INVASIVE TAMPERING ATTACKS

This section describes attacks in which the card is physically tampered with using special equipment. All micro probing techniques are invasive attacks. They require hours or weeks in a specialized laboratory and in the process they destroy the packaging of the card [6]. Invasive micro probing attacks require very little initial knowledge and usually work with a similar set of techniques on a wide range of products [6].

General Tampering Methodology

DE-PACKAGING

Before physical attacks can be performed, the circuit chip has to be removed from the plastic card. This can be done, by simply using a sharp knife to cut away the plastic, behind the chip module, until the epoxy resin becomes visible. Then, adding a few drops of fuming nitric acid can dissolve the resin. By shaking the card in acetone, until the silicon surface is fully exposed [3], the acid and resin can be washed away.

LAYOUT RECONSTRUCTION

After de-packaging, the next step in an invasive attack on a new processor is to create a map of it. The attacker uses an optical microscope with a CCD camera to produce large mosaics of high-resolution photographs of the chip surface. Basic structures such as data and address bus lines can be identified by studying connectivity patterns and by tracing metal lines that cross clearly visible module boundaries (ROM, RAM, EEPROM, ALU, instruction decoder, etc.) All processing modules are usually connected to the main bus via easily recognizable latches and bus drivers [6].

Deeper layers can only be recognized in a second series of photographs after the metal layers have been stripped off [6]. Images of successive layers of a chip can then be 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 [2123].

If the processor has a commonly accessible standard architecture, then the attacker has to reconstruct the layout only until he has identified those bus lines and functional modules that he has to manipulate in order to access all memory values [6].

At Cavendish laboratory in Cambridge, a technique has been developed for reverse engineering the circuit chips. The layout and function of the chip can be identified using this technique. Another technique developed by IBM can be used to observe the operation of the chip. As a result its secret keys can be fully revealed [3,21,23].

MANUAL MICRO-PROBING

The most important tool available for invasive attacks is a micro-probing workstation. Its major component is a special optical microscope. On an arm of the microscope, the attacker installs a probe, which is a metal shaft that holds a long tungsten-hair, which has been sharpened and allows the attacker to establish electrical contact with on-chip bus lines without damaging them. The probe is connected via an amplifier to a digital signal processor card that records or overrides processor signals and also provides the power, clock, reset, and I/O signals needed to operate the processor via pins [6].

USING ADVANCED BEAM TECHNOLOGIES

For future card generations with more metal layers and features below the wavelength of visible light, more expensive tools might have to be used in addition to the existing ones.

Ion Beams

A focused ion beam (FIB) workstation consists of a vacuum chamber with a particle gun. Gallium ions are accelerated and focused from a liquid metal cathode into a beam. Such beams can image samples from secondary particles [6,21]. Greater precision can be achieved by injecting a gas like iodine via a needle that is brought to within a few hundred micrometers from the beam target [6].

Further, by increasing the beam current, chip material can be removed with high resolution [6].

Today, attackers primarily to simplify manual probing of deep metal and poly-silicon lines use FIBs. A hole is drilled to the signal line of interest this is filled with platinum, to bring the signal to the surface, where a large probing pad, of several micrometers, or cross is created to allow easy access [6].

Electron Beam Testers

Other useful beam tools are electron-beam testers (EBT). EBTs are very convenient attack tools if the clock frequency of the observed processor can be reduced below 100 kHz, to allow real-time recording of all bus lines or if the processor can be forced to generate periodic signals, by continuously repeating the same transaction during the measurement [6].

Infrared Laser

A recently declassified technique invented at Sandia National Laboratories involves looking through the chip from the rear with an infrared laser using a wavelength at which the silicon substrate is transparent. The photocurrents thus

created allow probing of the device's operation and identification of logic states of individual transistors [21,23].

Attacks

KEY AND MEMORY READING

Reading ROM

While the ROM usually does not contain any cryptographic key material, it does often contain enough I/O, access control, and cryptographic routines to be of use in the design of a non-invasive attack [6].

Optical reconstruction techniques can be used to read ROM directly. The ROM bit pattern is stored in the diffusion layer, which leaves hardly any optical indication of the data on the chip surface [6]. Some ROM technologies store bits not in the shape of the active area but by modifying transistor threshold voltages. In this case, additional selective staining techniques have to be applied to make the bits visible [6].

Reading Memory Contents By Bus Probing

Except for ROM, it is usually not practical to read the information stored on a security processor directly out of each single memory cell. The stored data has to be accessed via the memory bus where all data is available at a single location. Micro-probing is used to observe the entire bus and record the values in memory as they are accessed [6].

Just replaying transactions might not suffice to make the processor access all critical memory locations. Sometimes, hostile bus observers are lucky and encounter a card where the programmer believed that by calculating and verifying some memory checksum after every reset the tamper-resistance could somehow be increased. This, of course, gives the attacker immediate easy access to all memory locations on the bus and considerably simplifies completing the read-out operation [6].

In order to read out all memory cells without the help of the card software, the attacker has to abuse a CPU component, such as an address counter, for it to access all memory cells for him. The program counter is already incremented automatically during every instruction cycle and used to read the next address, which makes it perfectly suited to serve as an address sequence generator. The attacker has to prevent the processor from executing jump, call, or return instructions, which would disturb the program counter in its normal read sequence. Tiny modifications of the instruction decoder or program counter circuit, which can easily be performed by opening the correct metal interconnect with a laser, often have the desired effect [6].

Reviving And Using The Test-Mode

Bovenlander has described (mentioned in [14]) breaking smartcards by using two microprobe needles to bridge the fuse blown at the end of the card test cycle, and using the re-enabled test routine to read out the memory contents.

Key Retrieval Using ROM Overwriting

Single bits in a ROM can be overwritten using a laser cutter microscope, and this ability can sometimes allow the attacker to make code changes that will lead to disclosure of the key.

A good example is with DES. Where the DES implementation is well known, the attacker can find one bit (or a small number of bits) with the property that by changing it/them will enable the key to be extracted easily. The details will depend on the exact implementation of DES but the attacker might be able, for example, to make a jump instruction unconditional and thus reduce the number of rounds to one or two. The attacker can also progressively remove instructions such as exclusive-or's of key material to make key extraction easier [14].

Alternatively, the DES S-boxes can be identified in the ROM and a number of their bits overwritten such that the encryption function becomes a linear transformation; the attacker can then extract the key from a single plaintext/ciphertext pair using linear cryptanalysis techniques [14].

ROM rewriting in general is also mentioned in very little detail in [19].

Key Retrieval Using EEPROM Overwriting

Keys can also sometimes be recovered if the attacker can modify the contents of EEPROM memory. A practical example, that was brought to notice in literature [14,26], is with DES. Suppose that the attacker knows the location of the DES key in EEPROM memory but cannot read it directly. He/she may still derive the key by modifying EEPROM contents. The attack is described below.

EEPROM rewriting is also mentioned in [19].

DES Key Retrieval Using Parity Checks

Anderson and Kuhn introduced an EEPROM modification attack. In their attack, an attacker is assumed to be able to write arbitrary values to arbitrary locations of the EEPROM, where the secret key is stored, but cannot read a value from the EEPROM. This is because the cost of writing a value to EEPROM is much lower than that of reading a value from EEPROM[1] [26]. This is a physical attack in which two micro-probing needles are used to set or clear target bits in order to infer those bits. If one bit of the secret key is set correctly, there would be no error in the output of the device; otherwise, error occurs. The secret key can thus be determined bit by bit [26]. In addition to requiring only low-cost equipment, this attack can be carried out with very few probing actions.

The attack in brief: Set the first bit of the EEPROM containing the target DES key to 1 (or 0, the choice doesn't matter) and operate the device. If it still works, the key bit was a 1. If you get a "key parity error" message[2], then the bit was zero. Move on to the next bit and so forth... [14]

[1] Writing can be done with low-cost equipment, such as microprobes, whereas reading requires much more expensive equipment, such as an electro-optical probe [26].

[2] Recall that the DES algorithm uses keys with odd parity, and a proper implementation will require that a key with the wrong parity will cause an error message to be returned.

Key Retrieval Using Gate Destruction

At least in DES, keys can be retrieved if the attacker has the ability to harm a gate in a register so it is stuck on a constant value throughout the cryptographic process.

DES is typically implemented with hardware for a single round, plus a register that holds the output of round *k* and sends it back as the input to round *k+1*. Biham and Shamir pointed out that if the least significant bit of this register is stuck, then the effect is that the least significant bit of the output of the round function is set to zero. By comparing the least significant six bits of the left half and the right half, several bits of key can be recovered; given about ten ciphertexts from a chip that has been damaged in this way, information about the round keys of previous rounds can be deduced using the techniques of differential cryptanalysis, and enough of the key can be recovered to make key-search easy [14].

According to [14], this attack is the first one that works against ciphers such as DES when the plaintext is completely unknown.

Key Retrieval Due To Memory Remanence

It has been shown that even RAM, that is supposed to lose its contents when the power is down, may physically keep portions of its contents for a while after it is disconnected from power, especially if the contents were there for a while. When values have been stored in computer memory for a long period of time, it is virtually impossible to erase them without leaving magnetic traces that can be used to recover the values. This is the basis of memory remanence attacks [19].

Gutman described the mechanisms that cause both static and dynamic RAM to "remember" values that they have stored for a long period of time [14].

The result is that although one might think that the protection mechanisms only have to deactivate the power supply, 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 (if exists) and reapply the power [21].

The remanence phenomenon also applies to the power-up state of memory. Long-term exposure to a constant bit pattern can cause some SRAM cells to adapt their "preferred" power-up state accordingly, an effect that can remain for several days without any supply voltage [21].

Key Retrieval By Probing Single Bus Bits

In [16] the authors show that by locally observing the value of a few RAM or address bus bits (possibly a single one) during the execution of a cryptographic algorithm, typically by the mean of a probing needle, an attacker could easily recover information on the secret key being used. The attacks shown in [16] apply to public-key cryptosystems such as RSA, as well as to secret-key encryption schemes including DES and RC5. The attacks are presented below.

This new kind of passive attack appears to be more powerful than the previous ones. No statistical analysis is needed in most cases. It is assumed that the attacker simply has access to a probe station, which briefly is a kind of needle

that allows monitoring the value of a single bit during the execution of some cryptographic algorithm.

The interesting feature of these attacks resides in that they are not necessarily destructive, as many previously suggested attacks are. In essence, probing does not always require the cutting of wires or inducing faults or even stressing the device to make it behave abnormally. For this attack the attacker just observes a single bit during execution.

Bus Probing for Asymmetric Algorithms Implementations

This attack method, shown in [16], completely recovers the exponent of a typical Square-and-Multiply implementation, thus providing a tool for breaking RSA, DSA, and others.

The model assumes that the attacker is provided with the value of certain accumulator bits at each execution of the internal loop of SM-1 (Square-Multiply). This means that the attacker collects bit-values just after the accumulator was squared or squared-then-multiplied.

The authors show how to infer *d* (the private key) by probing a single computation of the form $m^d \bmod n$ for given *m* and *n* when it is known that the exponentiation is done by SM-1[3]. A statistical analysis shows that the number of required guesses does not increase exponentially and the attack is indeed feasible.

If the attacker does not know the position of the bits that he probes, it is shown that he can guess them during the process itself.

Bus Probing for DES Implementations

According to [16], even a passive attacker may retrieve the secret key of a DES implementation given one single bit of information at each round.

Suppose an attacker uses an electronic station to locally observe the value of a given bit during the execution of DES. We assume that the attacker has sufficient knowledge of the device to be able to recognize two specific registers which are the *R* and *L* data registers on which the round function applies. Any bit of one or the other register is enough to attack the first and the last round subkeys. The article in [16] provides details of this attack, which are way beyond the scope of this document. It shall only be noted that the result is that 6 bits of the key are recovered, with 6 different ciphertexts, while running the attack.

Further, the same attack can be carried out on six bits of the secret key of the first round as well. So, a total of 12 bits can be recovered by this attack in total. The remaining 44 bits of the key can be found by exhaustive search.

Bus Probing for RC5 Implementations

Suppose that the attacker once again has access to all intermediate values of some bit *b* of either register *L* or register *R*, which is the case when he can probe one of these two in an iterated hardware implementation of the algorithm or a specific RAM buffer.

[3] Any attack in the above model could be sophisticated or generalized in various ways to support known variants of SM-1, such as right-to-left or multiple bit scanning.

The attack is detailed in [16] in several steps. The details are beyond the scope of the document. The results, however, are that the knowledge of a single intermediate bit, at each round, enables the attacker to derive the complete extended secret key and thus to further recover the initial secret key.

Complexity of this attack is actually very low and requires less than the exhaustive search of a single 32-bit subkey.

EEPROM ALTERATION

As all the key material of a smart card is stored in the electrically erasable, programmable, read only, memory (EEPROM), and due to the fact that EEPROM write operations can be affected by unusual voltages and temperatures, information can be trapped by raising or dropping the supplied voltage to the micro-controller [3]. In general, the big problem of EEPROM is that unusual voltage and temperatures can affect its write operations [18].

EEPROM Alteration By Voltage Changing

A widely known attack on the PIC16C84 micro-controller is that, by raising the voltage [3][4], the security bit of the controller can be cleared without erasing the memory. An attack on the DS5000 security processor is another example brought in [3]: A short voltage drop can release the security lock without always erasing the secret data[5].

Low voltage can facilitate other attacks as well, that are not related to EEPROM. An analogue random generator used to create cryptographic keys will produce an output of almost all 1's when the supply voltage is lowered slightly [3].

Moreover, erasing the charge stored in the floating gate of a memory cell requires a relatively high voltage. If the attacker can block this voltage from the card, changes might not be written [21].

A reader who is interested in knowing more on practical EEPROM attack using commonly available tools is encouraged to read [9].

EEPROM Alteration By UV Light

UV light can be focused on the security block cell of the EPROM to erase the lock bit, so data in the memory can be read [13,18].

[4] The method for doing so is also detailed in [13] and in [21].

[5] The method is also detailed in [13].

NON-INVASIVE AND ALGORITHM IMPLEMENTATION ATTACKS

Non-invasive attacks do not harm the card and are not card-specific. After the attacker has prepared such an attack for a specific processor type and software version, he can usually reproduce it within seconds on another card of the same type. The attacked card is not physically harmed and the equipment used in the attack can usually be disguised as a normal smartcard reader [6].

Non-invasive attacks are particularly dangerous in some applications for two reasons. Firstly, the owner of the compromised card might not notice that the secret keys have been stolen; therefore it is unlikely that the validity of the compromised keys will be revoked before they are abused. Secondly, non-invasive attacks often scale well, as the necessary equipment can usually be reproduced and updated at low cost [6].

Tamper evidence is of primary concern in applications such as banking and digital signatures, where the validity of keys can easily be revoked and where the owner of the card has already all the access that the keys provide anyway. Tamper resistance is also of importance in applications such as copyright enforcement, intellectual property protection, and some electronic cash schemes, where the security of an entire system collapses as soon as a few cards are compromised [6].

Unlike invasive probing attacks, the design of most non-invasive attacks requires detailed knowledge of both the processor and software [6].

Smartcard processors are particularly vulnerable to non-invasive attacks, because the attacker has full control over the power and clock supply lines. Larger security modules can be equipped with backup batteries, electromagnetic shielding, low-pass filters, and autonomous clock signal generators to reduce many of the risks to which smartcard processors are particularly exposed [6].

Timing Analysis Attacks

Timing attacks are based on measuring the time it takes for a unit to perform operations. This information can lead to information about the secret keys. For example: By carefully measuring the amount of time required to perform private key operations, an attacker might find fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems. If a unit is vulnerable, the attack is computationally simple and often requires only known ciphertext.

Cryptosystems often take slightly different amounts of time to process different inputs. Reasons include performance optimizations to bypass unnecessary operations, branching and conditional statements, RAM cache hits, processor instructions (such as multiplication and division) that run in non-fixed time, and a wide variety of other causes. Performance characteristics typically depend on both the encryption key and the input data (e.g., plaintext or ciphertext).

Attacks exist which can exploit timing measurements from vulnerable systems to find the entire secret key.

TIMING ATTACK ON RC5

A research document [2] describes a timing attack on the RC5 block encryption algorithm. The analysis is motivated by the possibility that some implementations of RC5 could result in the data-dependent rotations taking a time that is a function of the data.

Assuming that encryption timing measurements can be made which enable the cryptanalyst to deduce the total amount of rotations carried out during an RC5 encryption, it is shown that, for the nominal version of RC5, only a few thousand ciphertexts are required to determine 5 bits of the last half-round subkey with high probability. Further, it is shown that it is practical to determine the whole secret key with about 2^{20} encryption timings with a time complexity that can be as low as 2^{28}. [2]

The security of RC5 relies on the heavy use of data-dependent rotations. Kocher introduces the general notion of a timing attack. All implementations of RC5 on processors, which do not execute the rotation in constant time, are at risk from timing attacks. [2]

The authors of [2] have shown in some detail how to derive the extended secret key table of *RC5-32/12/16* by a timing attack using only about 2^{20} ciphertext timings and in time complexity 2^{28} in the best case, and 2^{40} in the worst case. This confirms Kocher's statement that RC5 is at some risk on platforms where rotations take a variable amount of time, and suggests one should be very careful when implementing RC5 on such platforms.

Power Consumption Attacks

Using a resistor in the power supply, we can measure with an analog/digital converter the fluctuations in the current consumed by the card. Drivers on the address and data bus often consist of up to a dozen parallel inverters per bit, each driving a large capacitive load.

Integrated circuits are built out of individual transistors, which act as voltage-controlled switches. Current flows across the transistor substrate when charge is applied to (or removed from) the gate. This current then delivers charge to the gates of other transistors, interconnect wires, and other circuit loads. The motion of electric charge consumes power and produces electromagnetic radiation, both of which are externally detectable [12].

The various instructions cause different levels of activity in the instruction decoder and arithmetic units and can often be quite clearly distinguished, so much so that parts of algorithms can be reconstructed [6].

SRAM write operations often generate the strongest signals. By averaging the current measurements of many repeated identical transactions, the attacker can even identify smaller signals that are not transmitted over the bus [6].

Signals such as carry bit states are of special interest, because many cryptographic key scheduling algorithms use shift operations that single out individual key bits in the carry flag. Even if the status-bit changes cannot be measured directly, they often cause changes in the instruction sequencer or micro-code execution, which then cause a clear change in the power consumption [6].

More background information on power analysis (SPA, DPA and HO-DPA) can also be found in [27].

SIMPLE POWER ANALYSIS (SPA)

Simple Power Analysis (SPA) is a technique that involves directly interpreting power consumption measurements collected during cryptographic operations. SPA can yield information about a device's operation as well as key material [7]. Because SPA can reveal the sequence of instructions executed, it can be used to break cryptographic implementations in which the execution path depends on the data being processed [7].

In SPA attacks, an attacker directly observes a system's power consumption. The amount of power consumed varies depending on the microprocessor instruction performed. Large features such as DES rounds, RSA operations, etc. may be identified, since the operations performed by the microprocessor vary significantly during different parts of these operations. At higher magnification, individual instructions can be differentiated. SPA analysis can, for example, be used to break RSA implementations by revealing differences between multiplication and squaring operations. Similarly, many DES implementations have visible differences within permutations and shifts [12].

SPA on DES Key schedule

The DES key schedule computation involves rotating 28-bit key registers. A conditional branch is commonly used to check the bit shifted off the end so that *"1"* bits can be wrapped around. The resulting power consumption traces for a *"1"* bit and a *"0"* bit will contain different SPA features if the execution paths take different branches for each [7].

Moreover, DES implementations perform a variety of bit permutations. Conditional branching in software or micro-code can cause significant power consumption differences for *"0"* and *"1"* bits [7].

SPA on Comparison Operations

String or memory comparison operations typically perform a conditional branch when a mismatch is found. This conditional branching causes large SPA characteristics [7].

SPA on Multipliers Operation

Modular multiplication circuits tend to leak a great deal of information about the data they process. The leakage functions depend on the multiplier design, but are often strongly correlated to operand values and hamming weights [7].

SPA on Exponentiators Operation

A simple modular exponentiation function scans across the exponent, performing a squaring operation in every iteration with an additional multiplication operation for each exponent bit that is equal to *"1"*. The exponent can be compromised if squaring and multiplication operations have different power consumption

characteristics, take different amounts of time, or are separated by different code. Modular exponentiation functions that operate on two or more exponent bits at a time may have more complex leakage functions [7].

DIFFERENTIAL POWER ANALYSIS (DPA)

In addition to large-scale power variations, due to the instruction sequence, there are effects correlated to data values being manipulated. These variations tend to be smaller and are sometimes overshadowed by measurement errors and other noise. In such cases, it is still often possible to break the system using statistical functions tailored to the target algorithm [7]. While SPA attacks primarily use visual inspection to identify relevant power fluctuations, DPA attacks use statistical analysis and error correction techniques to extract information correlated to secret keys [12].

While some products can withstand simple power analysis, the authors of [12] have not found any commercially available products that resist DPA.

For good background information on Differential Power Analysis (DPA) the reader is encouraged to read the DPA Q&A in [8].

DPA of Asymmetric Algorithms Implementations

Public key algorithms can be analyzed using DPA by correlating candidate values for computation intermediates with power consumption measurements. For modular exponentiation operations, it is possible to test exponent bit guesses by testing whether predicted intermediate values are correlated to the actual computation. Chinese Remainder Theorem (CRT) RSA implementations can also be analyzed, for example by defining selection functions over the CRT reduction or recombination processes [7].

Because of the relatively high computational complexity of multiplication operations [7], signals leaking during asymmetric operations tend to be much stronger than those from many symmetric algorithms.

HIGH ORDER DIFFERENTIAL POWER ANALYSIS (HO-DPA)

While the DPA techniques described above analyze information across a single event between samples, high-order DPA may be used to correlate information between multiple cryptographic sub-operations [12].

In a high-order DPA attack, signals collected from multiple sources, signals collected using different measuring techniques, and signals with different temporal offsets are combined during application of DPA techniques. Additionally, more general differential functions may be applied, as well as more advanced signal processing functions [12].

According to [12], no actual systems are known that are vulnerable to High-Order DPA that are not also vulnerable to "regular" DPA. However, DPA countermeasures must also address HO-DPA attacks to be fully effective.

Differential Fault Analysis (DFA)

Differential fault analysis (DFA) is a powerful attack on crypto systems embodied in devices such as smart cards. If the device can be made to deliver erroneous output under stress (heat, vibration, pressure, radiation, whatever) then a cryptanalyst comparing correct & erroneous outputs has a dangerous entry point to the processors internals, including keys [5].

Differential Fault Analysis can break many secret key cryptosystems, including DES, IDEA, RC5 and Feal [4].

In [14] it is claimed that attacks based on inducing errors in instruction code are easier, and more informative, than attacks based on inducing errors in data. Therefore, it is claimed that introducing program execution glitches is more effective than forcing errors in data.

DFA Attack On DES Implementations

In [4], Biham and Shamir describe a DFA attack and show that it is applicable to almost any secret key cryptosystem proposed so far in the open literature. They have actually implemented DFA in the case of DES, and demonstrated that under the same hardware fault model used by the Bellcore researchers in 1996 (where attack on public key algorithms was shown), they can extract the full DES key from a sealed tamperproof DES machine by analyzing fewer than 200 ciphertexts generated from unknown cleartexts.

The power of Differential Fault Analysis is further demonstrated by the fact that even if DES is replaced by triple DES (whose 168 bits of key were assumed to make it practically invulnerable), essentially the same attack can break it with essentially the same number of given ciphertexts [4].

The attack in [4] follows the Bellcore fundamental assumption that by exposing a sealed tamperproof device such as a smart card to certain physical effects (e.g., ionizing or microwave radiation), one can induce with reasonable probability a fault at a random bit location in one of the registers at some random intermediate stage in the cryptographic computation. Both the bit location and the round number in which the error occurred are unknown to the attacker.

The authors of [4] have implemented this attack on a personal computer. They found the whole last subkey given less than 200 ciphertexts, with random single-faults in all the rounds.

The complete detail of the attack is beyond the scope of this document (it can be found in [4]). The attack is used to find the last subkey. Once this subkey is known, the attacker can proceed in one of two ways: He can use the fact that this subkey contains 48 out of the 56 key bits in order to guess the missing 8 bits in all the possible 2^8=256 combinations. Alternatively, he can use the knowledge of the last subkey to peel up the last round (and remove faults that we already identified), and analyze the preceding rounds with the same data using the same attack. This latter approach makes it possible to attack triple DES (with 168 bit keys) as well, or DES with independent subkeys (with 768 bit keys).

Some criticism of this attack can be found at [14]. According to the author of [14], the problem with these proposed attacks is that no one has demonstrated the feasibility of the fault model itself. With many security processors, the key material is held in EEPROM together with several kilobytes of executable code; so it is likely that a random one-bit error which did have an effect on the device's behavior would be more likely to crash the processor (by overwriting a part of the code) or yield an uninformative error than to produce a faulty ciphertext of the kind required for the above attacks. In my opinion, even if this is true, the lower chance of attack success does not eliminate the need to combat this type of attack. The attack is worrisome even if circumstantial conditions cause it to fail naturally in most (but not all) cases.

DFA Attack On Public Key Algorithms Implementations

Again, we assume that by exposing a sealed tamperproof device such as a smart card to certain physical effects (e.g., ionizing or microwave radiation), one can induce, with reasonable probability, faults at random bit locations in a tamperproof device at some random intermediate stage in the cryptographic computation. It is further assumed that the attacker is in physical possession of the tamperproof device and that he can repeat the experiment with the same private key by applying external physical effects to obtain outputs due to faults [15].

The following two attacks show that one bit fault at certain location and time can cause fatal leakage of the secret key. The attacks are presented in [15] and are also discussed in [25].

Attack #1

The following specific attack is presented in [15].

Suppose that one bit in the binary representation of d (the private exponent) is changed from 1 to 0 or vice versa, and that the faulty bit position is randomly located. An attacker arbitrarily chooses a plaintext M and computes the ciphertext C. He then applies external physical effects to the tamperproof device and at the same time asks the device to decrypt C. Assuming that d(i) is changed to its complement d(i)', then the output of the device will be M'= C(t-1)^d(t- 1))(C(t-2)^d(t-2))...(C(i)^d(i)') ...(C(1)^d(1))(C(0)^d(0)) mod n. Since he now possesses both M and M', he can compute M'/M = C(i)^d(i)'/C(i)^d(i) mod n. Obviously, if M'/M = 1/C(i) mod n, then d(i) = 1, and if M'/M = C(i) mod n, then d(i) = 0. The attacker can pre-compute C(i) and 1/C(i) mod n for i = 0, 1, ..., t-1, and compares M'/M mod n to these values in order to determine one bit of d. He repeats the above process using either the same plaintext-ciphertext pair or using different plaintext-ciphertext pairs until he finds enough information to obtain d.

It should be noted that the attack also works for multiple bit errors. Details can be found in [15].

The general RSA attack concept, that is presented, can also be applied to attacking discrete logarithm based public key cryptosystems [15].

Attack #2

The following specific attack is presented in [15].

Suppose that the one bit error is in C(i), we denote the corrupted value as C(i)'. Then the output from the tamperproof device is M' =(C(t-1)^d(t-1))(C(t- 2)^d(t-2))...(C(i)'^d(i)) ...(C(1)^d(1))(C(0)^d(0))mod n. The attacker computes M'/M = C(i)'^d(i)/C(i)^d(i) = C(i)'/C(i) mod n (note that d(i) in this ratio must be 1). He can compute all the possible C(i)'/C(i) mod n values in advance (there are a total of t^2 such values) and store them somewhere. Now the attacker compares all these values with M'/M mod n. Once a match is found, he knows i and then knows that d(i) is 1. The above procedure is repeated until enough information is obtained to determine d.

It should be noted that the attack also works for multiple bit errors. Details can be found in [15].

Glitch Attacks

In a glitch attack, the attacker deliberately generates a malfunction that causes one or more flip-flops to adopt the wrong state. The aim is usually to replace a single critical machine instruction with an almost arbitrary other one. Glitches can also aim to corrupt data values as they are transferred between registers and memory [6].

I am aware of three techniques for creating fairly reliable malfunctions that affect only a very small number of machine cycles in smartcard processors: clock signal transients, power supply transients, and external electrical field transients.

Particularly interesting instructions, that an attacker might want to replace with glitches, are conditional jumps or the test instructions preceding them. They create a window of vulnerability in the processing stages of many security applications that often allow the attacker to bypass sophisticated cryptographic barriers by simply preventing the execution of the code that detects that an authentication attempt was unsuccessful. Instruction glitches can also be used to extend the runtime of loops, for instance in serial port output routines, to see more of the memory after the output buffer, or also to reduce the runtime of loops, for instance to transform an iterated cipher function into an easy to break single-round variant [6].

Clock-signal glitches are currently the simplest and most practical ones. They temporarily increase the clock frequency for one or more half cycles, such that some flip-flops sample their input before the new state has reached them [6].

Power analysis can also be used to monitor how far a program has progressed. This in turn can be used to determine when, for example, a branch instruction is about to be taken. A more rapid clock cycle at this point (a clock glitch) may provide insufficient time for the processor to write the jump address to the program counter, thereby annulling the branch operation [11].

A similar clock-glitch attack is also presented in [14]. The idea is to apply a glitch in either the clock or the power supply to the chip. Because of the different number of gate delays in various signal paths and the varying parameters of the circuits on the chip, this affects only some signals, and by varying the precise timing and duration of the glitch, the CPU can be made to execute a number of completely different, wrong instructions. These will vary from one instance of the chip to another, but can be found by a systematic search using simple hardware.

SPECIFIC GLITCH ATTACK ON RSA IMPLEMENTATION

If a smartcard computes an RSA signature *S* on a message *M* modulo $n = p*q$ by computing it *mod p* and *mod q* separately and then combining them using the Chinese Remainder Theorem (CRT), and if an error can be induced in either of the former computations, then the attacker can factor *n* at once.

Since the card spends most of its time calculating the signature *mod p* and *mod q*, and almost any glitch that affects the output will do, the attacker does not have to be selective about where in the instruction sequence the glitch is applied. Since only a single signature is needed, the attack can even be performed online. Such an attack is explained in [14] as well as in [15] and in [25].

Another related clock-glitch attack against RSA and DES was done by Ross and Kuhn and is presented in [19].

SPECIFIC GLITCH ATTACK ON DES IMPLEMENTATION

When we can cause an instruction of our choice to fail, then there are several fairly straightforward ways to attack DES. We can remove one of the 8-bit *xor* operations that are used to combine the round keys with the inputs to the S-boxes from the last two rounds of the cipher, and repeat this for each of these key bytes in turn. The erroneous ciphertext outputs that we receive as a result of this attack will each differ from the genuine ciphertext in the output of usually two, and sometimes three, S-boxes. Using the techniques of differential cryptanalysis, we obtain about 5 bits of information about the 8 key bits that were not xor'ed as a result of the induced fault. So, for example, 6 ciphertexts with faulty last rounds should give us about 30 bits of the key, leaving an easy brute-force search [14].

An even faster attack presented in [14] is to reduce the number of rounds in DES to one or two by corrupting the appropriate loop variable or conditional jump.

To sum it up: DES can be fully compromised with somewhere between one and ten faulty ciphertexts.

Another clock-glitch attack against RSA and DES was done by Ross and Kuhn and is presented in [19].

SPECIFIC GLITCH ATTACK ON RC5 IMPLEMENTATION

Apart from its key schedule, RC5 may be about the worst possible algorithm choice for secret-algorithm hardware applications, where some implementations may be vulnerable to glitch attacks [14]. Details of the attacks against RC5 are beyond the scope of this document. RC5 is vulnerable to glitches by design of the specific cipher.

About Discretix

Discretix is a semiconductor intellectual property company that develops and licenses advanced embedded security solutions for resource-constrained environments, such as wireless devices and smart-cards, where stringent limits apply to the cost, size and power consumption of the target devices.

Discretix technology has already been adopted by some of the major vendors of wireless baseband and application chipset, as well as smart-card IC vendors.

Discretix Technologies Ltd.

Corporate Headquarters
43 Hamelacha Street
Beit Etgarim
Poleg Industrial Zone
Netanya 42504
Israel
Tel: +972 9 885 8810
Fax: +972 9 885 8820
Email:
marketing@discretix.com

Representative in Japan:
Triangle Technologies KK
Sogo-Hirakawacho Bldg.
4F 1-4-12
Hirakawacho Chiyoda-ku
Tokyo, Japan
Te l: +81 3 5215 8760
Fax: +81 3 5215 8765
Email:
japan.info@discretix.com