SMART CARD CONTENT SECURITY

Defining "tamperproof" for portable smart media

AUTHOR: Stefano Zanero (s.zanero@computer.org)
Dipartimento di Elettronica e Informazione
Politecnico di Milano

Version 1.0

ABSTRACT

Smart Cards are often touted as "secure" portable storage devices. A complete, high-level design metodology has been proposed for embedded information systems based on smart card devices. However, this metodology takes as granted that informations stored on the card will be really securely stored, and access control will be correctly maintained. Unfortunately, standards and specifications, created by hardware and software vendors for both the card hardware and the micro operating system which runs it have been repeatedly proven not as secure as they are commonly supposed to be.

In this paper we try to analyze the faults in existing standards and implementations of content security for smart card embedded information systems, and we try to suggest possible ways (both hardware and software) to prevent security leaks. This paper does not provide breaking news, but rather tries to sum up the known techniquest to attack smart card devices.

1 SMART CARD CONCEPTS

1.1 CARD TYPES. WHAT IS SMART ?

The International Organization for Standardization (ISO) standard 7810[1] "Identification Cards – Physical Characteristics" defines physical properties such as flexibility, temperature resistance, and dimensions for three different card formats (ID-1, ID-2, and ID-3).

There are different types of ID-1 format cards, each specified by a different substandard[2]:

Embossed cards: embossing allows for textual information or designs on the card to be transferred to paper by using a simple and inexpensive device. ISO 7811[3] specifies the embossed marks, covering their form, size, embossing height, and positioning. Transfer of information via embossing may seem primitive, but the simplicity of the system has made worldwide proliferation possible.

Magnetic Stripe: the primary advantage that magnetic stripe technology offers over embossing is a reduction in the flood of paper documents. Parts 2, 4, and 5 of ISO 7811 specify the properties of the magnetic stripe, coding techniques, and positioning. The stripe's storage capacity is about 1000 bits and anyone with the appropriate read/write device can view or alter the data.

Integrated Circuit cards (smart cards): these are the newest and most clever additions to the ID-1 family, and they also follow the details laid down in the ISO 7816[4] series. These types of cards allow far greater orders of magnitude in terms of data storage – cards with over 20 Kbytes of memory are currently available. Also, and perhaps most important, the stored data can be protected against unauthorized access and tampering. Memory functions such as reading, writing, and erasing can be linked to specific conditions, controlled by both hardware and software. Another advantage of smartcards over magnetic stripe cards is that they are more reliable and have longer expected lifetimes.

Memory Cards: though often also referred to as smartcards, memory cards are typically much less expensive and much less functional than microprocessor cards. They contain EEPROM and ROM memory, as well as some address and security logic. In the simplest designs, logic exists to prevent writing and erasing of the data. More complex designs allow for memory read access to be restricted. Since they cannot directly manipulate data they are dependent on the card reader (also known as the card-accepting device) for their processing and are suitable for uses where the card performs a fixed operation. Typical memory card applications are pre-paid telephone cards and health insurance cards.

Contactless Smartcards: though the reliability of smartcard contacts has improved to very acceptable levels over the years, contacts are one of the most frequent failure points any electromechanical system due to dirt, wear, etc. The contactless card solves this problem and also provides the issuer an interesting range of new possibilities during use. Cards need no longer be inserted into a reader, which could improve end user acceptance. No chip contacts are visible on the surface of the card so that card graphics can express more freedom. Still, despite these benefits, contactless cards have not yet seen wide acceptance. The cost is higher and not enough experience has been gained to make the technology reliable. Nevertheless, this elegant solution will likely have its day in the sun at some time in the future.

Optical Memory Cards: ISO/IEC standards 11693[5] and 11694[6] define standards for optical memory cards. These cards look like a card with a piece of a CD glued on top - which is basically what they are. They can carry many megabytes of data, but can only be written once and never erased with today's technology. Today, these cards have no processor in them (although this is coming in the near future). While the cards are comparable in price to chip cards, the card read and write devices use non-standard protocols and are still very expensive. However such cards may find use in applications such as health care where large amounts of data must be stored.

	Maximum memory capacity (nominal)	Type of on-board CPU	Card cost	Cost of reader, software, connections
Magnetic-stripe cards	140 bytes	None	$0.20 - $0.75	$750
Integrated circuit memory cards	1 Kbyte	None	$1 - $2.50	$500
Integrated circuit processor cards ("Smart cards")	8 Kbytes	8-bit CPU (16 o 32 bit in the near future)	$7-$15	$500
Optical Memory Cards	2.8 - 4.9 Mbyte	None	$7 - $12	$3,500 - $4,000

1.2 SMART CARD BASICS

Integrated Circuit Cards have conventionally come to be known as "Smart cards". A smart card is a card that is embedded with either a microprocessor and a memory chip or only a memory chip with non-programmable logic. As we will see, this simple and somehow strange structure offers a bunch of functionalities difficult to obtain otherwise.

The microprocessor card can add, delete, and otherwise manipulate information on the card, while a memory-chip card (for example, pre-paid phone cards) can only undertake a pre-defined operation.

Smart cards, unlike magnetic stripe cards, can carry all necessary functions and information on the card. Therefore, they do not require access to remote databases at the time of the transaction.

A typical smartcard consists of an 8-bit microprocessor running at approximately 5 MHz with ROM, EEPROM and RAM, together with serial input and output, all in a single chip that is mounted on a plastic carrier. The operating system is typically stored in ROM, the CPU uses RAM as its working memory, and most of the data is stored in EEPROM.

A rule of thumb for smartcard silicon is that RAM requires four times as much space as EEPROM, which in turn requires four times as much space as ROM. There are various smart card chipset. The most common chipsets mount 32 kbytes of ROM, and either 32 kbytes of EEPROM with 1 Kbyte RAM or 16 Kbytes of EEPROM with 2 Kbytes of RAM. This gives them the equivalent processing power of the original IBM-XT computer, albeit with slightly less memory capacity.

In addition, most smart cards embed a cryptographic coprocessor. Because the common asymmetric cryptographic algorithms of the day (such as RSA) require very large integer math calculations, an 8 bit microprocessor with very little RAM can take on the order of several minutes to perform a 1024 bit private key operation. However, if a cryptographic coprocessor is added to the architecture, the time required for this same operation is reduced to around a few hundred microseconds. The coprocessors include additional arithmetic units developed specifically for large integer math and fast exponentiation. There is a drawback, however, and it is the cost. The addition of a cryptographic coprocessor can increase the cost of today's smartcards by 50% to 100%. These cost increases will likely diminish as coprocessors become more widespread.

Smart cards are passive devices, which means that to function a smart card needs to be inserted into a reader connected to a computer, or an integrated smart terminal. These devices are usually known as CAD (Card Acceptance Device), and come in many kind of shapes: readers integrated into a vending machine, handheld battery-operated readers with a small LCD screen, readers integrated into a GSM mobile phone, or attached to a personal computer by a variety of interfaces. Mechanically, readers have various options including: whether the user must insert/remove the card versus automated insertion/ejection mechanism, sliding contacts versus landing contacts, and provisions for displays and keystroke entry. Electrically, the reader must conform to the ISO/IEC 7816-3[4] standard.

The CAD offers power for the smartcard chip, and an interface for communication, which is bidirectional and half-duplex (one-way at a time). The serial I/O interface usually consists of a single register, through which the data is transferred in a half duplex manner, bit by bit. Though the chip can be thought of as a tiny computer, the external terminal must supply the voltage, ground, and clock. It could also be important to remember that, though commonly referred to as "smartcard readers", all smartcard enabled terminals, by definition, have the ability to read and write as long as the smartcard supports it and the proper access conditions have been fulfilled

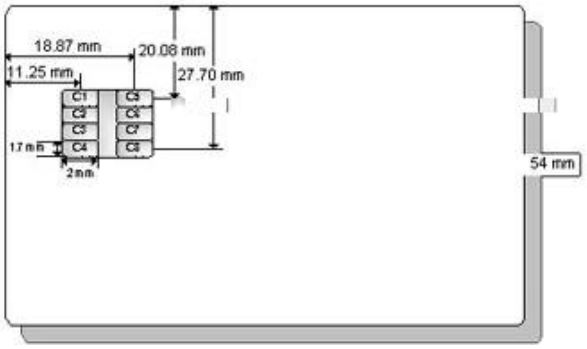
There are standards for data transfer format, CAD specifications, and chipset interface specifications. You may refer to ISO 7816[4] standard (based on ID-1 type cards, as specified by ISO7810 standard), which has originated ETSI, EMV and Open Card[7] standards. These standards have been widely adopted, leading to interoperability of various cards and products.

A smart card works in a black-box model: the CAD gives the card an input, this input is processed by the card chipset, and then an output is sent back to the CAD. The CAD itself cannot access directly the smart card EEPROM, RAM or ROM memories.

Since data cannot be retrieved directly via the CAD, smart cards have been proposed as portable and secure data storage devices. In addition, their computing capabilities (expecially if integrated by the cryptogaphic co-processor) make them expecially suitable as private key storage devices for asymetric algorithms, since in this way private keys can be generated and stored on board the card, and never leave it. Encription and decription of data are performed on request by the card chipset itself. In this way, the user's private key is kept secure and can not be eavesdropped. Thus, chip cards have been the main platform for holding a secure digital identity.

1.3 PHYSICAL AND ELECTRICAL PROPERTIES

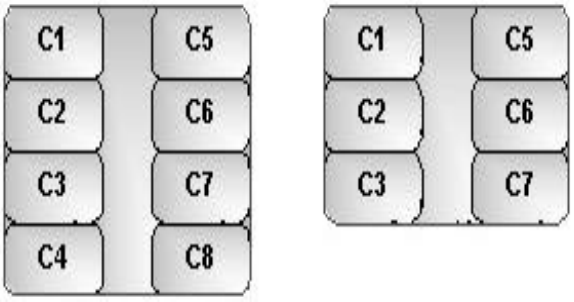
As we already said, the physical size and shape of a smartcard is described in ISO 7810[1] and designated as ID-1. The dimensions are 85.6 mm by 54 mm, with a corner radius of 3.18 mm and a thickness of 0.76mm. ISO 7810[1] was created in 1985, so it did not address chip placement but instead addressed embossing, magnetic stripes, and so on. Smartcard chip placement is defined in ISO 7816-2[4], which is dated 1988. See figure for details:



Card robustness requirements are specified in ISO 7810[1], 7813[8], and 7816[4] part 1. These specifications address such things as UV radiation, X-ray radiation, the card's surface profile, mechanical robustness of card and contacts, electromagnetic susceptibility, electromagnetic discharges, and temperature resistance. ISO/IEC 10373[9] specifies the test methods for many of these requirements.

The electrical specifications for smartcards are defined in ISO/IEC 7816 parts 2 and 3, and GSM 11.11[10]. Most smartcards have eight contact fields on the front face, however, two of these are reserved for future use so some manufacturers produce cards with only six contact fields, which slightly reduces production costs.

Electrical contacts are typically numbered C1 through C8 from top left to bottom right, as shown here both for 6 and 8 contact shapes:



In the table we list for each contact a standard abbreviation and a short function description:

POSITION	ABBREV.	FUNCTION
C1	Vcc	Supply Voltage
C2	RST	Reset
C3	CLK	Clock Frequency
C4	RFU	Reserved for future use
C5	GND	Ground
C6	Vpp	External programming voltage
C7	I/O	Serial input/output communications
C8	RFU	Reserved for future use

The Vpp contact was used several years ago to supply voltage to EEPROMs for programming and erasing. However, with the advent of charge pumps that exist on the chip, the Vpp contact is rarely used today (see below for security implications of this change). The Vcc supply voltage is specified at 5 volts ± 10%. There is an industry push for smartcard standards to support 3 volt technology because all mobile phone components are available in a 3 volt configuration, and smartcards are the only remaining component which require a mobile phone to have a charge converter. It is theoretically possible to develop 3-volt smartcards, but interoperability with current 5-volt systems would be a problem. Nonetheless, a wider voltage range handling 3 to 5 volts will probably become mandatory in the near future.

1.4 DATA TRANSMISSIONS

All communications to and from the smartcard are carried out over the C7 contact. Thus, only one party can communicate at a time, whether it is the card or the terminal. This is termed "half-duplex". Communication is always initiated by the terminal, which implies a type of client/server relationship between card and terminal.

After a card is inserted into a terminal, it is powered up by the terminal, executes a power-on-reset, and sends an Answer to Reset (ATR) to the terminal. The ATR is parsed, various parameters are extracted, and the terminal then submits the initial instruction to the card. The card generates a reply

and sends it back to the terminal. The client/server relationship continues in this manner until processing is completed and the card is removed from the terminal.

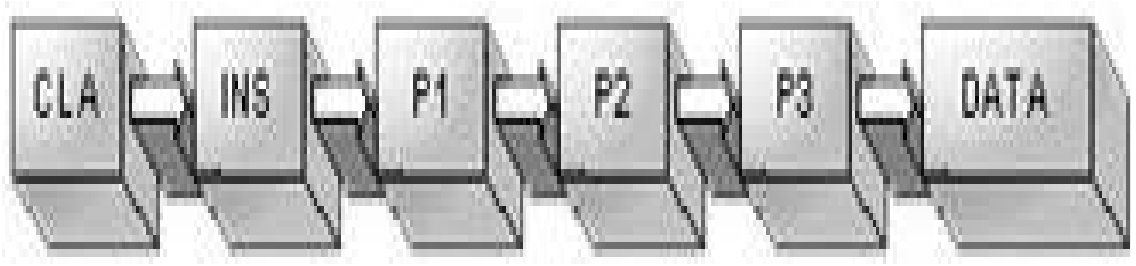
The physical transmission layer is defined in ISO/IEC 7816-3. It defines the voltage level specifics which end up translating into the "0" and "1" bits.

Logically, there are several different protocols for exchanging information in the client/server relationship. They are designated "T=" plus a number, as summarized here:

PROTOCOL	DESCRIPTION
T = 0	Asynchronous, half-duplex, byte oriented, see ISO/IEC 7816-3
T = 1	Asynchronous, half-duplex, block oriented, see ISO/IEC 7816-3, Adm.1
T = 2	Asynchronous, full-duplex, block oriented, see ISO/IEC 10536-4
T = 3	Full duplex, not yet covered
T = 4	Asynchronous, half-duplex, byte oriented, (expansion of T = 0)
T = 5 TO T = 13	Reserved for future use
T = 14	For national functions, no ISO standard
T = 15	Reserved for future use

The two protocols most commonly seen are T=0 and T=1, T=0 being the most popular. A brief overview of the T=0 protocol is given below. The references contain more detailed information and descriptions of all the protocols.

In the T=0 protocol, the terminal initiates communications by sending a 5 byte instruction header which includes a class byte (CLA), an instruction byte (INS), and three parameter bytes (P1, P2, and P3). This is followed optionally by a data section.



Most commands are either incoming or outgoing from the card’s perspective and the P3 byte specifies the length of the data that will be incoming or outgoing. Error checking is handled exclusively by a parity bit appended to each transmitted byte. If the card correctly receives the 5 bytes, it will return a one-byte acknowledgment equivalent to the received INS byte.

If the terminal is sending more data (incoming command) it will send the number of bytes it specified in P3. Now the card has received the complete instruction and can process it and generate a response. All commands have a two-byte response code, SW1 and SW2, which reports success or an error condition. If a successful command must return additional bytes, the number of bytes is specified in the SW2 byte.

In this case, the GET RESPONSE command is used, which is itself a 5-byte instruction conforming to the protocol. In the GET RESPONSE instruction, P3 will be equal to the number of bytes specified in the previous SW2 byte. GET RESPONSE is an outgoing command from the card’s

point of view. The terminal and card communicate in this manner, using incoming or outgoing commands, until processing is complete.

1.5 SMART CARD OPERATING SYSTEMS

There's a wide variety of operating systems designed for smart cards. They suffer most limitations common for embedded operating systems, in particular for size and performance. The size is typically between 3 and 24 Kbytes. The lower limit is that used by specialized applications and the upper limit by multi-application operating systems.

Though typically only a few thousand bytes of program code, the operating system for the smartcard microprocessor must handle such tasks as:

- ?? Data transmission over the bi-directional, serial terminal interface
- ?? Loading, operating, and management of applications
- ?? Execution control and Instruction processing
- ?? Protected access to data
- ?? Memory Management
- ?? File Management
- ?? Management and Execution of cryptographic algorithms

Just like embedded operating systems, they do not need user interfaces or the ability to access external peripherals or storage media.

There are four international standards that define typical smartcard instruction sets. More than 50 instructions and their corresponding execution parameters are defined. Though found in four separate standards, the instructions are largely compatible. The specifications are GSM 11.11 (prETS 300608), EN 726-3[11], ISO/IEC 7816-4, and the preliminary CEN standard prEN 1546[12].

Instructions can be classified by function as follows:

- ?? File selection
- ?? File reading and writing
- ?? File searching
- ?? File operations
- ?? Identification
- ?? Authentication
- ?? Cryptographic functions
- ?? File management
- ?? Instructions for electronic purses or credit cards
- ?? Operating system completion
- ?? Hardware testing
- ?? Special instructions for specific applications
- ?? Transmission protocol support

Because smartcard memory space is so severely limited, not all standardized instructions and file structures can be generally implemented in all smartcard operating systems. For this reason, so-called "Profiles" have been introduced in ISO 7816-4 and EN 726-3. A profile defines the minimum requirements for data structures and commands.

For example, Profile O in ISO 7816-4 defines the following minimums:

Data Structures:	Transparent
	Linear Fixed
	Linear Variable
	Cyclic
Commands:	READ BINARY, UPDATE BINARY, no implicit selection and maximum length up to 256 bytes
	READ RECORD, UPDATE RECORD, without automatic selection
	APPEND RECORD
	SELECT FILE
	VERIFY
	INTERNAL AUTHENTICATE
	EXTERNAL AUTHENTICATE
	GET CHALLENGE

1.5.1 JAVA CARDS

One of the most common smart card operating environments (adopted by over the 95% of manufacturers) is Java. Java-enabled smart cards are called Java Cards[13]. A complete discussion of the Java Card architecture is far beyond the scopes of this work. However, we will discuss it briefly to give an example of how a smart card OS could implement access to card databanks and access controls.

Just as in the Java operating environment for computer systems, the JavaCard API enables a "Write Once, Run Anywhere" approach, by wrapping proprietary, vendor-dependant API and system calls into a common framework.

The Java programming language and the Java Card API allow development using modern object-oriented programming, instead of assembly language or the C programming language. Using OOP has obvious benefits for security, allowing the developer to encapsulate sensitive data and algorithms within objects, which have provable behaviour and are easier to test; this is obviously in addition to traditional benefits for time-to-market and maintainability.

In addition, the Java community has developed a wide and strong base of knowledge on the security and safety issue, which can be leveraged when developing smart-card applications.

As an additional security benefit the Java Card platform provides a secure execution environment with a "firewall" (beware: not in the traditional meaning) between different applications in the same card. This allows different applications on the same card to function separately and independently from each other as if they were on separate cards. We will see that this is a benefit against software-based attack.

In the last five years, products incorporating the Java Card platform have passed real-world security evaluations for major industries around the world. The Java Card platform is the leading platform for multi-application cards in mobile telephony. It is also the only platform that has passed security evaluations for issuance by all major financial payment associations. In addition, it has passed security assessments by leading government authorities, including the US Department of Defense and the US National Security Agency. Java Card platforms have achieved compliance with FIPS 140-1.

1.6 CRYPTOGRAPHIC CAPABILITIES

Current state of the art smartcards have sufficient cryptographic capabilities to support popular security applications and protocols. In spite of the increased cost, the benefits to computer and network security of including the cryptographic coprocessor are great, for it allows the private key never to leave the smartcard. As we'll see in the following sections, this becomes a critical factor for operations such as digital signatures, authentication, and non-repudiation. Eventually, though, the need for a cryptographic coprocessor and its associated cost will likely go away. The basic processors could become powerful enough to perform the math-intensive operations, or other algorithms such as those based on elliptic curve technology could become popular. Elliptic curve algorithms provide strong security without the need for large integer math, but haven't yet found their way into widespread use.

However, we will better describe common capabilities found in the crypto-enabled smartcards from leading vendors.

RSA signatures and verifications are supported with a choice of 512, 768, or 1024 bit keylengths. The algorithms typically use the Chinese Remainder Theorem (CRT) in order to speed up the processing. Even at the 1024 bit keylength, the time needed to perform a signature is typically under one second. Usually the EEPROM file that contains the private key is designed such that the sensitive key material never leaves the chip. Even the card holder can't access the key material in this case. The usage of the private key is protected by the user's PIN, so that possession of the card does not imply the ability to sign with the card. RSA's PKCS#1 padding is implemented by some cards.

Though smartcards have the ability to generate RSA keypairs, this can be very slow. Typical times needed for a 1024 bit RSA keypair range from 8 seconds to 3 minutes. The larger times violate the ISO specifications for communications timeout so specialized hardware or software is sometimes necessary. Also, the quality of the keypairs may not be extremely high. The lack of computing power implies a relatively weak random number source as well as relatively weak algorithms for selecting large prime numbers.

The Digital Signature Algorithm (DSA) is less widely implemented than RSA. When it is implemented, it is typically found only at the 512 bit key length.

DES and triple DES are commonly found in the leading smartcards. They usually have the option to be used in a Message Authentication Code (MAC) function. However, because the serial interface of a smartcard has a low bandwidth, bulk symmetric encryption is very slow.

Electronic purse functionalities are often present, but they are typically based on symmetric key technologies such as DES and triple DES. Thus, a shared secret key enforces the security of many of these schemes. Hashing algorithms commonly found include SHA-1 and MD-5; but again the low bandwidth serial connection hinders effective use of bulk hashing on the card.

Random number generation (RNG) varies among card vendors. Some implement a pseudo RNG where each card has a unique seed. In this case, random numbers cycle through, dependent on the algorithm and the seed. Some cards have a true, hardware based RNG using some physical aspect of the silicon. It's best to check with the vendor for details of the RNG if it will be used in a cryptographically sensitive context.

As with any technology, there are legal issues to keep in mind when dealing with smartcards. Commonly, a smartcard has the ability to perform certain licensed algorithms, such as the RSA asymmetric cipher. Usually any license fees associated with the algorithm are bundled into the cost of the smartcard.

1.7 SECURITY FEATURES

We already saw that one of the basics concepts on which smart card security architecture relays is that it should be really difficult to extract informations about card operating and file systems from the device without controls by both the chip and the card OS. To do so, various methods of hardware security monitoring are enabled on leading smartcards.

A one-time, irreversible fuse typically disables any test code built into the EEPROM. In order to avoid card cloning an unalterable serial number is often burned into the memory. The cards are designed to reset themselves to a power-on state if they detect fluctuations in voltage, temperature, or clock frequency. Reading or Writing of the ROM is usually disabled. However, since every vendor has its own, usually proprietary, schemes for these measures, it's always good to inquire and/or request reports from independent testing laboratories.

Communications protocols on smartcards at the command level can also have a security protocol built in. These are typically based on symmetric key technology and allow the smartcard itself to authenticate the read/write terminal or vice versa. However, the cryptograms and algorithms for these protocols are usually specific to a given application and terminal set.

Smartcards support the ability to configure multiple PINs that can have different purposes. Applications can configure one PIN to be a "Security Officer" PIN, which can unblock the User PIN, after a set number of bad PIN attempts, or re-initialize the card. Other PINs can be configured to control access to sensitive files or purse functions.

2 SMART CARD USAGE

2.1 EXAMPLES OF SMART CARD USAGE

Since data stored on a smart card cannot be retrieved directly via the CAD, smart cards have been proposed as portable and secure data storage devices. In addition, their computing capabilities (expecially if integrated by the cryptographic co-processor) make them expecially suitable as private key storage devices for asymmetric algorithms, since in this way private keys can be generated and stored on board the card, and never leave it. Encryption and decryption of data are performed on request by the card chipset itself. In this way, the user's private key is kept secure and can not be eavesdropped. Thus, chip cards have been the main platform for holding a secure digital identity.

Smart Cards are now everywhere: in GSM phones (the SIM, Subscriber Identity Module, is a smart card), in new generation credit cards, in pay-TV and digital satellite decoders, and as a personal

data holder in the next-generation of ID card projects. They are also used for credit cards and prepaid phone cards. Combining their two main functions of being a secure data container and a crypto-enabled device, cards can:

- ?? securely hold money ("stored value cards") or money equivalents
- ?? provide secure access to a network, secure identification, law-strong digital signature
- ?? secure cellular phones from fraud
- ?? allow set-top boxes on televisions to remain secure from piracy

Even though smartcards provide many obvious benefits to computer security, they still haven't caught on with great popularity in countries like the United States. This is not only because of the prevalence, infrastructure, and acceptability of magnetic stripe cards, but also because of a few problems associated with smartcards.

Lack of infrastructure for smartcard reader/writers is often cited as a complaint. The major computer manufactures haven't until very recently given much thought to offering a smartcard reader as a standard component. Many companies don't want to absorb the cost of outfitting computers with smartcard readers until the economies of scale drive down their cost. In the meantime, many vendors provide bundled solutions to outfit any personal computer with smartcard capabilities.

Lack of widely adopted smartcard standards is often cited as a complaint. The number of smartcard related standards is high and many of them address only a certain vertical market or only a certain layer of communications. This problem is lessening recently as web browsers and other mainstream applications are including smartcards as an option. Applications like these are helping to speed up the evolution of standards.

2.2 SMART CARD AS SECURITY TOKENS

2.2.1 USING SMART CARD AS CRYPTO DEVICES

Smart Cards are extraordinarily useful as crypto devices. A primary reason for this is that they have the quite unique ability of being capable of generating and protecting a private signing key which can never leave the card. In this way it is really difficult for outsiders to gain knowledge of the private key, something which could otherwise happen for example through a compromise of the host computer system. This has obvious and immediate advantages on protocols and applications oriented to authentication, authorization, privacy, integrity, and non-repudiation, for example *PKI*, Public Key Infrastructure, systems. These systems offer the services listed above by the means of a public/private key asymmetric algorithm. Now, placing the private certificate on a smartcard, which it never leaves, the crucial secret for the system is never in a situation where it is easily compromised. Moreover, if a private key is stored in a browser storage file on a hard drive, it is typically protected by a password. This file can be "dictionary attacked" where commonly used passwords are attempted in a brute force manner until knowledge of the private key is obtained. On the other hand, a smartcard will typically lock itself up after some low number of consecutive bad PIN attempts, for example 10. Thus, the dictionary attack is no longer a feasible way to access the private key if it has been securely stored on a smartcard.

In addition, wherever multiple disjointed systems often have their security based on different technologies, smartcards can bring these together by storing multiple certificates and passwords on the same card. One of the biggest problems in typical password systems is that users write down their password and attach it to their monitor or keyboard. They also tend to choose weak passwords

and share their passwords with other people. If a smartcard is used to store a user's multiple passwords, they need only remember the PIN to the smartcard (and to own the device, of course) in order to access all of their passwords, which at this point can be really strong, random alphanumeric strings. The end user need never even know the passwords, so that they can't be written down or shared with others.

In addition to ease of use, the so called "two factor" authentication is stronger by nature than one factor. For authenticating yourself, there are basically 3 methods: something you know (a shared secret, or password), something you have (a token), or something you are (biometrics). A smart card is two factor, since it needs both something you know (the PIN) and something you have (the card itself). There are also prototypes of CADs with an integrated fingerprint scanner, which constitute a robust three factor authentication.

2.2.2 SECURITY-RELATED STANDARDS

This section discusses the principles of the most prominent standards that are used to integrate smartcard into computer applications to provide security related services. Any standard designed to facilitate the integration of smartcards into computer security systems should follow certain principles in order to be useful and gain acceptance:

- **Multi-platform:** any standard should be applicable to the whole wide variety of modern day operating systems and computer architectures;
- **Open participation:** any standard should be formed and reviewed through input and peer review from members of industry, academia, and government;
- **Interoperability:** any standard should be interoperable with other leading standards and protocols;
- **Functional:** any standard should apply to real world problems and markets and adequately address their requirements;
- **Experience:** any standard should be created by a group of people with real-world experience in security-related products and standards;
- **Extensibility:** any standard should facilitate expansion to new applications, protocols, and smartcard capabilities that weren't yet around when the standard was created.

The following are emerging as important standards with respect to the integration of smartcards into computer and network security applications:

- **PKCS#11: Cryptographic Token Interface Standard**[14]
  This standard specifies an Application Programming Interface (API), called Cryptoki, to devices which hold cryptographic information and perform cryptographic functions. Cryptoki, pronounced crypto-key and short for cryptographic token interface, follows a simple object-based approach, addressing the goals of technology independence (any kind of device) and resource sharing (multiple applications accessing multiple devices). PKCS#11 presents to applications a common, logical view of the device called a cryptographic token. The standard was created in 1994 by RSA with input from industry, academia, and government.
- **PC/SC**
  The PC/SC Workgroup[15] was formed in May 1997. It was created to address critical technical issues related to the integration of smartcards with the PC. PC/SC Workgroup members include Bull Personal Transaction Systems, Gemplus, Hewlett-Packard, IBM, Microsoft Corp., Schlumberger, Siemens-Nixdorf Inc., Sun Microsystems, Toshiba Corp., and VeriFone. The specification addresses limitations in existing standards that complicate

integration of ICC devices with the PC and fail to adequately address interoperability, from a PC application perspective, between products from multiple vendors. It provides standardize interfaces to Interface Devices (IFDs) and the specification of common PC programming interfaces and control mechanisms. Version 1.0 was released in December of 1997.

?? **OpenCard**[7]
OpenCard is a standard framework announced by International Business Machines Corporation, Inc., Netscape, NCI, and Sun Microsystems Inc. that provides for interoperable smartcard solutions across many hardware and software platforms. The OpenCard Framework is an open standard providing an architecture and a set of APIs that enable application developers and service providers to build and deploy smartcard aware solutions in any OpenCard-compliant environment. It was first announced March, 1997.

?? **JavaCard**[13]
The JavaCard API is a specification that enables the Write Once, Run Anywhere capabilities of Java on smartcards and other devices with limited memory. The JavaCard API was developed in conjunction with leading members of the smart card industry and has been adopted by over 95% of the manufacturers in the smart card industry, including Bull/CP8, Dallas Semiconductor, De La Rue, Geisecke & Devrient, Gemplus, Inside Technologies, Motorola, Oberthur, Schlumberger, and Toshiba.

?? **Common Data Security Architecture**[16]
Developed by Intel, the Common Data Security Architecture (CDSA) provides an open, interoperable, extensible, and cross-platform software framework that makes computer platforms more secure for all applications including electronic commerce, communications, and digital content. The CDSA 2.0 specifications were adopted by The Open Group in December 1997.

?? **Microsoft Cryptographic API**[17]
The Microsoft® Cryptographic API (CryptoAPI) provides services that enable application developers to add cryptography and certificate management functionality to their Win32® applications. Applications can use the functions in CryptoAPI without knowing anything about the underlying implementation, in much the same way that an application can use a graphics library without knowing anything about the particular graphics hardware configuration.

2.2.3 APPLICATION EXAMPLES

?? **Web Browsers (SSL, TLS)**
Web browsers use technology such as Secure Sockets Layer (SSL) and Transport Layer Security (TLS) to provide security while browsing the World Wide Web. These technologies can authenticate the client and/or server to each other and also provide an encrypted channel for any message traffic or file transfer. The authentication is enhanced because the private key is stored securely on the smartcard. The encrypted channel typically uses a symmetric cipher where the encryption is performed in the host computer because of the low data transfer speeds to and from the smartcard. Nonetheless, the randomly generated session key that is used for symmetric encryption is wrapped with the partner's public key, meaning that it can only be unwrapped on the smartcard. Thus it is very difficult for an eavesdropper to gain knowledge of the session key and message traffic.

?? **Secure Email (S/MIME, OpenPGP)**
S/MIME and OpenPGP allow for email to be encrypted and/or digitally signed. As with SSL, smartcards enhance the security of these operations by protecting the secrecy of the private key and also unwrapping session keys within a security perimeter.

?? **Form Signing**
Web based HTML forms can be digitally signed by your private key. This could prove to be a very important technology for internet based business because it allows for digital documents to be hosted by web servers and accessed by web browsers in a paperless fashion. Online expense reports, W-4 forms, purchase requests, and group insurance forms are some examples. For form signing, smartcards provide portability of the private key and certificate as well as hardware strength non repudiation.

?? **Object Signing**
If an organization writes code that can be downloaded over the web and then executed on client computers, it is best to sign that code so the clients can be sure it indeed came from a reputable source. Smartcards can be used by the signing organization so the private key can't be compromised by a rogue organization in order to impersonate the valid one.

?? **Kiosk / Portable Preferences**
Certain applications operate best in a "kiosk mode" where one computer is shared by a number of users but becomes configured to their preferences when they insert their smartcard. The station can then be used for secure email, web browsing, etc. and the private key would never leave the smartcard into the environment of the kiosk computer. The kiosk can even be configured to accept no mouse or keyboard input until an authorized user inserts the proper smartcard and supplies the proper PIN.

?? **File Encryption**
Even though the 9600 baud serial interface of the smartcard usually prevents it from being a convenient mechanism for bulk file encryption, it can enhance the security of this function. If a different, random session key is used for each file to be encrypted, the bulk encryption can be performed in the host computer system at fast speeds and the session key can then be wrapped by the smartcard. Then, the only way to easily decrypt the file is by possessing the proper smartcard and submitting the proper PIN so that the session key can be unwrapped.

?? **Workstation Logon**
Logon credentials can be securely stored on a smartcard. The normal login mechanism of the workstation, which usually prompts for a username and password, can be replaced with one that communicates to the smartcard.

?? **Dialup Access (RAS, PPTP, RADIUS, TACACS)**
Many of the common remote access dial-up protocols use passwords as their security mechanism. As previously discussed, smartcards enhance the security of passwords. Also, as many of these protocols evolve to support public key based systems, smartcards can be used to increase the security and portability of the private key and certificate.

?? **Payment Protocols (SET)**
The Secure Electronic Transactions (SET) protocol allows for credit card data to be transferred securely between customer, merchant, and issuer. Because SET relies on public key technology, smartcards are a good choice for storage of the certificate and private key.

?? **Digital Cash**
Smartcards can implement protocols whereby digital cash can be carried around on a smartcard. In these systems, the underlying keys that secure the architecture never leave the security perimeter of hardware devices. Mondex[18], VisaCash[19], EMV (Europay-Mastercard-Visa), and Proton[20] are examples of digital cash protocols designed for use with smartcards.

?? **Building Access**
Even though the insertion, processing time, and removal of a standard smartcard could be a hassle when entering a building, magnetic stripe or proximity chip technology can be added to smartcards so that a single token provides computer security and physical access.

?? **Law-strong digital signatures**
New digital signature laws are being written by many states that make it the end user's responsibility to protect their private key. If the private key can never leave an automatically PIN disabling smartcard, then the end user can find it easier to meet these responsibilities. Certificate authorities can help in this area by supporting certificate extensions that specify the private key was generated in a secure environment and has never left the confines of a smartcard. With this mechanism, higher levels of non-repudiation can be achieved when verifying a smartcard based signature while using a certificate containing such an extension. In other words, a digital signature carries more weight if its associated certificate validates that the private key resides on a smartcard and can never be extracted.

2.3 SECURITY EVALUATION OF SMART CARDS

2.3.1 SECURITY DESIGN STANDARDS

The ultimate goal of smart card security is proven robustness and correct functioning of every single card delivered to the card user. Chip security and card life cycle security are the key links in this chain. Chip and card life cycle security are non-competitive issues which means that these properties should not and cannot be separated in the design process.

The market for smart cards is highly cost sensitive; differences of a few cents per card matter when millions of units are involved. This means that any defensive measures must meet very stringent cost effectiveness tests that are unusual with other IT products. Attacks that involve multiple parts of a security system are difficult to predict and model. If cipher designers, software developers, and hardware engineers do not understand or review each other's work, security assumptions made at each level of a system's design may be incomplete or unrealistic. As a result, security faults often involve unanticipated interactions between components designed by different people. For example, National Institute of Standard and Technology (NIST) emphasizes the importance of computer security awareness and of making information security a management priority that is communicated to all employees[21].

2.3.2 SMART CARD SECURITY EVALUATION

Currently, Financial Payment Systems, i.e. credit card brands, individually do smart card evaluations – unstandardized, possibly conflicting[22]. Vendor's products may be subject to conflicting requirements, repeated and expensive evaluations by different users. ISO 15408 – Common Criteria for Information Technology Security Evaluation, the "CC", represents the outcome of efforts to develop criteria for evaluation of IT security that are widely useful within the international community. It is an alignment and development of a number of source criteria: The existing European, US, and Canadian criteria (ITSEC, TCSEC and CTCPEC respectively). The Common Criteria resolves the conceptual and technical differences between the source criteria. It is a contribution to the development of an international standard, and opens the way to worldwide mutual recognition of evaluation results. Version 1.0 of the CC was published for comment in January 1996. Version 2.1, the current version, was published in December 1999. If independent third party evaluation should become mandatory, it would require sharing test methods and information about vulnerabilities between private companies and independent institutions[23]. A public acceptance of an evaluation scheme could even require an open discussion and disclosure of

information about risks and vulnerabilities to the public. It is therefore unfortunate if smart card security really depends on confidentiality of CPU design and specifications.

Common Criteria established an handful of important concepts in security system evaluations:

- There should be a common structure and language for expressing product or system IT security requirements[24]
- There should be "catalogs" of standardized IT security requirement components and packages. The CC presents requirements[25] for the IT security of a product or system under the distinct categories of functional requirements and assurance requirements. The CC functional requirements define desired security behavior. Assurance requirements are the basis for gaining confidence that the claimed security measures are effective and implemented correctly.
- The CC envisages the definition of Protection Profile (PP), standardized and well understood sets of implementation independent security requirements developed by a user group to specify their security functionality needs for a particular product (there are examples in literature[26]). This allows a manufacturer or product developer to build a product according to the requirements of a PP. They can then have it evaluated and claim conformance to the PP. The product is still evaluated against a security target (ST) but the contents of the ST mirror the requirements laid down in the PP. A security target is created by the product vendor and is therefore implementation specific.

The smart card protection profile presented in this study is a joint effort of the Smart Card Security User Group (SCSUG). SCSUG is a global financially oriented industry group formed specifically to represent the security needs of the user community. It comprises of American Express, Europay, JCB, MasterCard, Mondex, Visa, NIST and NSA.

As most readers surely know, before Common Criteria development one of the most accepted security standards was ITSEC (which served as a basis for the CC themselves). A study[27] gives us an overview how certification work under ITSEC and Common Criteria schemes. It points out the difficulty of comparison of these two schemes. There are several issues which favor CC over ITSEC.

The advantage of the second important concept in CC is that the security functionality will be expressed in an explicit, unambiguous way. The wording is well understood and includes detailed guidance for interpretation and application. The first important concept in CC makes comparison of certifications by users and mutual recognition by certification bodies more practical. The detailed guidance in CC on calculating attack potential aims at removing some of the subjectivity from this difficult assessment task and it may offer more clarity than the ITSEC. The Smart Card Security User Group protection profile emphasizes that a vulnerability to certain types of threats can only be ascertained by examining the IC, operating system and applications as an integrated whole because effective security relies on a synergistic contribution of these three layers.

It was further noted in the same study that all the examined ITSEC certifications claimed a high Strength of Mechanisms (SoM) but the scope of each evaluation was also limited in some way, either to particular phases of the card life cycle, by exclusion of the chip from the Target of Evaluation or by specifically excluding relevant threats. It can be questioned whether a high SoM would have been attained if all threats were considered in the context of the integrated product, as it is issued to the user in its actual mode of use.

3 ATTACKS TO SMART CARDS

3.1 OUR APPROACH

As we already said, smart cards promise numerous security benefits. Unlike magnetic stripe cards, they can protect the stored data against unauthorized access. However, the strength of this protection seems to be frequently overestimated.

There is no security system which is unbreakable[28], or at least no one has designed such a system until now. Designing secure systems is a matter of balancing costs and benefits, which means it is truly a matter of engineering. Availability, Integrity, and Confidentiality, can be only "partially" granted, and time and resources devoted to these requirements must be correctly balanced against other functional requirements.

In this perspective, analyzing the security of a system means wondering if it properly protects the value of the information stored inside, or: how much would it cost (measured in time, in money, in skill and in effort) to a given attacker to execute successfully an attack? And how much benefit could he have? As engineers and designers we must be conscious of the existence of attacks, and deploy countermeasures as appropriate, until we are reasonably sure that an attack would be far too costly for an attacker than any benefit he could obtain.

A simple example of design gone awry is the protection scheme used for digital content protection on satellite TV channels. Almost every existing scheme has been cracked, but in some cases the hack is so simple and immediate, and requires so small knowledge and skill, that almost anyone can perform it directly with a simple PC and a smart card programmer. If, for example, decoding a single transmission required an hour of preparation, no one would loose so much time.

A taxonomy of attackers in three classes of increasingly high danger has been proposed[29] by IBM researchers as follows:
Class I (clever outsiders): They are often very intelligent but may have insufficient knowledge of the system. They may have access to only moderately sophisticated equipment. They often try to take advantage of an existing weakness in the system, rather than try to create one.
Class II (knowledgeable insiders): They have substantial specialized technical education and experience. They have varying degrees of understanding of parts of the system but potential access to most of it. They often have highly sophisticated tools and instruments for analysis.
Class III (funded organisations): They are able to assemble teams of specialists with related and complementary skills backed by great funding resources. They are capable of in-depth analysis of the system, designing sophisticated attacks, and using the most advanced analysis tools. They may use Class II adversaries as part of the attack team.

Obviously various attacks will or won't be available to any of these classes of attackers, and any device we design can be made arbitrarily secure. We must understand which kind of attackers we will be facing before attempting to design the security of a smart card system (or any security system, in fact). Only with this information available we will be able to make proper design and engineering decisions.

As we said in the abstract, in this paper we will try to list most of the known and well-documented methods to attack smart-card based systems. We will focus on attacks against the smart-card itself or its interaction with the CAD device. We will also briefly discuss API and OS level attacks. We will not deal with protocols and applications relaying on smart cards for security, or with the issues associated with digital signature, non repudiation or authentication schemes. We will focus on the

security of the smart card itself, and of its contents. A complete taxonomy of all the kinds of security glitches and attacks that a card may suffer, along with countermeasures, has been proposed by Bruce Schneier to a USENIX conference[30].

Independent security labs test for common security attacks on leading smartcards, and can usually provide an estimate of the cost in equipment and expertise of breaking the smartcard. When choosing a smartcard for an architecture, one can ask the manufacturer for references to independent labs that have done security testing. Using this information, designers can strive to ensure that the cost of breaking the system would be much greater than the value of any information obtained.

3.2 INVASIVE ATTACK TECHNIQUES

3.2.1 GENERALITIES

An attack on a smart card is defined "invasive" if it involves such a tampering of the device which is clearly visible for anyone. In fact, most of the techniques listed here require an utter destruction of the card hardware.

In addition, while non-invasive attacks can usually be performed by "borrowing" for some small amount of time a smart card device, invasive attacks can require hours of work in specialized labs and are therefore available only to highly skilled and funded attackers.

Therefore there is a small probability that such an attack could be performed without knowledge of the user (who will realize, sooner or later, that he no longer owns his card), and there is also small point in performing such attacks on authentication and signature devices (which can be revoked when the user discovers the loss).

However, if we are designing a smart card based information system which stores highly valuable information on board the card, such attacks are clearly a concern.

3.2.2 REMOVING THE CHIP FROM THE CARD

All invasive attacks starts with the removal of the chip package. As we said before, the typical chip module consists of a thin plastic basis plate of about a square centimetre with conductive contact areas on both sides. One side is visible on the final card and makes contact with the card reader; the silicon die is glued to the other side, and connected using thin gold or aluminium bonding wires. The chip side of the plastic plate is then covered with epoxy resin. The resulting chip module is finally glued into the card.

Removing the chip is easy. First, we use a sharp knife or hand lathe to cut away the plastic behind the chip module until the epoxy resin becomes visible; otherwise we may just heat the card plastic until it becomes flexible, and then remove the chip by simply bending the card.

Now we cover the chip with 20-50 ml of hot (60°C) fuming nitric acid (>98% HNO_3) on the resin and wait a few minutes until the acid dissolves the black epoxy resin that encapsulates the silicon drive. Then we wash acid and resin away by shaking the card in acetone in an ultrasonic bath. We can repeat this procedure if necessary, and conclude it with a final bath in deionised water and isopropanol[31].

3.2.3 REVERSE ENGINEERING OF THE CHIPSET

If the chip itself is custom-designed to perform special functions, there is a nonzero probability that the designers violated the Kerckhoff's Principle, which states that the strength of a crypto device – and by extension of a security device – must rely solely into the secret, or key, not into the structures and the algorithms.

In other words, if they felt that their custom design was "secret", its secrecy could well be its only protection. Security through obscurity has never worked, nor does it work with smart cards. Card components can be reverse engineered. Some steps of this process even apply to cards with standard processors – you will still need to understand the layout of bus lines and chip modules on the chipset, even if you know perfectly well what each one does.

The first step is to create a map of a new processor. It could be done by using an optical microscope with a CCD camera to produce meters-large, high-resolution photographs of the chip surface. The attacker has to be familiar with CMOS VLSI design techniques and microcontroller architectures, but the necessary knowledge is easily available from numerous textbooks[32] [33] [34] [35]. Deeper layers can only be recognized in a second series of photographs after the metal layers have been stripped off, which can be achieved by etching the chip, for instance by submerging it for a few seconds in hydrofluoric acid (HF) in an ultrasonic bath[31]. HF quickly dissolves the silicon oxide around the metal tracks and detaches them from the chip surface. This is called "wet etching". Details on how to examine circuits, on tools and methods, are present in literature[36].

More sophisticated tools like focused ion beam (FIB) workstations can be used to perform attacks. A focused ion beam (FIB) workstation consists of a vacuum chamber with a particle gun, comparable to a scanning electron microscope (SEM). Gallium ions are accelerated and focused from a liquid metal cathode with 30 kV into a beam of down to 5-10 nm diameter, with beam currents ranging from 1 pA to 10 nA. FIBs can image samples from secondary particles similar to a SEM with down to 5 nm resolution.

By increasing the beam current, chip material can be removed[37] with the same resolution at a rate of around 0.25 $\mu m^3\ nA^{-1}\ s^{-1}$. Better etch rates 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. Gas molecules settle down on the chip surface and react with removed material to form a volatile compound that can be pumped away and is not redeposited. Using this gas-assisted etch technique, holes that are up to 12 times deeper than wide can be created at arbitrary angles to get access to deep metal layers without damaging nearby structures. By injecting a platinum-based organo-metallic gas that is broken down on the chip surface by the ion beam, platinum can be deposited to establish new contacts. With other gas chemistries, even insulators can be deposited to establish surface contacts to deep metal without contacting any covering layers.

Using laser interferometer stages, a FIB operator can navigate blindly on a chip surface with 0.15 µm precision, even if the chip has been planarized and has no recognizable surface structures. Chips can also be polished from the back side down to a thickness of just a few tens of micrometers. Using laser-interferometer navigation or infrared laser imaging, it is then possible to locate individual transistors and contact them through the silicon substrate by FIB editing a suitable hole. This rear-access technique has probably not yet been used by pirates so far, but the technique is about to become much more commonly available and therefore has to be taken into account by designers of new security chips.

FIBs are used by attackers today primarily to simplify manual probing of deep metal and polysilicon lines. A hole is drilled to the signal line of interest, filled with platinum to bring the signal to the surface, where a several micrometer large probing pad or cross is created to allow easy access. Modern FIB workstations cost about half a million euro and are available in over hundred organizations. Processing time can be rented from numerous companies all over the world for a few hundred euro per hour.

Another useful particle beam tool are electron-beam testers (EBT)[38]. These are SEMs with a voltage-contrast function. Typical acceleration voltages and beam currents for the primary electrons are 2.5 kV and 5 nA. The number and energy of secondary electrons are an indication of the local electric field on the chip surface and signal lines can be observed with submicrometer resolution. The signal generated during EBT is essentially the low-pass filtered product of the beam current multiplied with a function of the signal voltage, plus noise. EBTs can measure waveforms with a bandwidth of several gigahertz, but only with periodic signals where stroboscopic techniques and periodic averaging can be used. If we use real-time voltage-contrast mode, where the beam is continuously directed to a single spot and the blurred and noisy stream of secondary electrons is recorded, then the signal bandwidth is limited to a few megahertz[38]. While such a bandwidth might just be sufficient for observing a single signal line in a 3.5 MHz smartcard, it is too low to observe an entire bus with a sample frequency of several megahertz for each line.

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. Therefore, a low-frequency alarm is commonly found on smartcard processors. However, simple high-pass or low-pass RC elements are not sufficient, because by carefully varying the duty cycle of the clock signal, we can often prevent the activation of such detectors. A good low-frequency sensor must trigger if no clock edge has been seen for longer than some specified time limit (e.g., 0.5 µs).

In this case, the processor must not only be reset immediately, but all bus lines and registers also have to be grounded quickly, as otherwise the values on them would remain visible sufficiently long for a voltage-contrast scan.

Even such carefully designed low-frequency detectors can quite easily be disabled by laser cutting or FIB editing the RC element. To prevent such simple tampering, an article[47] suggests that an intrinsic self-test can be built into the detector, so that any attempt to tamper with the sensor should result in the malfunction of the entire processor. The authors have designed such a circuit that tests the sensor during a required step in the normal reset sequence. External resets are not directly forwarded to the internal reset lines, but only cause an additional frequency divider to reduce the clock signal. This then activates the low-frequency detector, which then activates the internal reset lines, which finally deactivate the divider. The processor has now passed the sensor test and can start normal operation.

The processor is designed such that it will not run after a power up without a proper internal reset. A large number of FIB edits would be necessary to make the processor operational without the frequency sensor being active. Other sensor defenses against invasive attacks should equally be embedded into the normal operation of the processor, or they will easily be circumvented by merely destroying their signal or power supply connections.

Another article[39] details how at the Cambridge University microelectronics lab, they were able to build an apparatus for smart card reverse engineering, consisting of a slightly modified electron

beam lithography machine (this functions in effect as an electron microscope) and a PC with an image processing system (a DCT chip and locally written software). They then developed techniques for etching away a layer at a time without doing too much damage. Conventional wet etching causes too much havoc with half micron chips, so “dry etching” is used in which gases such as CF4 or HF strip off layers of silica and aluminium in turn.

One of their innovations is a technique to show up N and P doped layers in electron micrographs. This uses the Schottky effect: a thin film of a metal such as gold or palladium is deposited on the chip creating a diode effect which can be seen with the electron beam.

Finally, image processing software has been developed to spot the common chip features and reduce the initially fuzzy image of the metal tracks into a clean polygon representation. There are also routines to get images of successive layers, and of adjacent parts of the chip, in register.

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

This is typical of the kind of attack which an academic lab can mount. Even more sophisticated attacks, invented at Sandia National laboratories and recently published[40], involve looking through the chip. Light-Induced Voltage Alteration is a non-destructive technique that involves probing operating ICs from the back side with an infrared laser to 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. Low-Energy Charge Induced Voltage Alteration relies on a surface interaction phenomenon that produces a negative charge-polarization wave using a low- energy electron beam generated by a scanning electron microscope. This allows imaging the chip to identify open conductors and voltage levels without damage, although it does not operate through metallization layers. Of course, even more sophisticated techniques may be available in classified government facilities.

As a defense, a number of copy trap features are incorporated into commercial chip designs. For example, we have heard of design elements that look like a transistor, but are in reality only a connection between gate and source; and 3-input NORs which function only as 2-input NORs. Many of these copier traps are based on holes in isolating layers or on tricks done in the diffusion layer with ion implantation (based on the assumption that it is hard to distinguish N from P). However the layer etching and Schottky techniques developed by Haroun Ahmed's team can detect such traps.

Another possibility is to introduce complexity into the chip layout and to use nonstandard cell libraries. However the chip still has to work, which limits the complexity; and nonstandard cells can be reconstructed at the gate level and incorporated in the recognition software. Finally, in the Clipper chip (created by U.S. Government and NSA as a standard black box encryption module) there are a number of silicon features, of which the most important is a fusible link system. These links are only fused after fabrication and hold the long term key and other secret aspects of the chip. Details can of course be found in a paper in the relevant data book[41], and from the scanning electron micrographs there, it is clear that the secret information can be recovered by sectioning the chip. This technique has been used by Professor Ahmed's team on occasion on obscure features in other chips.

Thus the effect of current silicon level copy traps is just to slow down the attacker. In fact, there are widespread rumors that Intel has reverse engineered the Clipper chip, but that the results have been classified. However, a successful attack[42] against the protocols (and not the hardware) of the Clipper chip made further research virtually uninteresting.

The same appears to be the case for chemical measures. Chips intended for classified military use are often protected by passivation layers of a tenacity never encountered in civilian packaging[43]. But here again, informed sources agree that with enough effort, techniques can be developed to remove them.

We understand that neither silicon copy traps not advanced passivation techniques are used by smartcard manufacturers in the bulk of their products. The marketing director of a smartcard manufacturer said that they simply had no demand from their users for anything really sophisticated. The most that appears to be done is an optical sensor under an opaque coating.

Hi-tech techniques may indeed have been used by commercial pirates to duplicate satellite TV smartcards. Recent postings to a TV hackers' mailing list recount how an undergraduate used nitric acid and acetone to remove ICs intact from Sky-TV smartcards; he then put them in the University's electron beam tester (an ICT 8020, also sold as the Advantest E 1340 - a 1991 machine). The chips were run in a test loop, but he had been unable to remove the silicon nitride passivation layer; the many secondary electrons removed from this caused it to get charged positive very quickly, which obscured the underlying circuit. He did not have access to a dry etching facility to remove this layer, and could get no further. However it is significant that a person with no funding or specialist knowledge could get even this far.

3.2.4 MICROPROBING

Microprobing means removing the chip from the card and interacting directly with its components. It is closely associated with reverse engineering of the chip (in fact, we separated the paragraph for a purely didactical purpose). By microprobing, we violate the black box assumption (that is, the card can be accessed only through a proper CAD) and therefore cause almost all the protection scheme to fail (since it’s based fundamentally on that one assumption).

A microprobing attack starts with the removal of the chip package, but even once the chip is opened, it is still not possible to perform probing or modification attacks: to do so, you need to remove at least part of the passivation layer (which is a layer of silicon nitride or oxide, which protects them from environmental influences and ion migration) before probes can establish contact. This is not affected by nitric acid; chip testers typically remove it using dry etching with hydrogen fluoride, a process that is not as easily performed by amateur hackers.

But dry etching is not the only option. Another approach is to use microprobing needles that remove the passivation just below the probe contact point using ultrasonic vibration. Laser cutter microscopes commonly used in cellular biology laboratories have also been used to remove the passivation locally. The UV or green laser is mounted on the camera port of the microscope and fires laser pulses through the microscope onto rectangular areas of the chip with micrometer precision. Carefully dosed laser ashes remove patches of the passivation layer. The resulting hole in the passivation layer can be made so small that only a single bus line is exposed. This prevents accidental contacts with neighbouring lines and the hole also stabilizes the position of the probe and makes it less sensitive to vibrations and temperature changes.

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

The most important tool for invasive attacks is a microprobing workstation. Its major component is a special optical microscope (e.g., Mitutoyo FS-60) with a working distance of at least 8 mm between the chip surface and the objective lens. On a stable platform around a socket for the test package, we install several micropositioners, which allow us to move a probe arm with submicrometer precision over a chip surface. On this arm, we install a "cat whisker" probe. This is a metal shaft that holds a 10 µm diameter and 5 mm long tungsten-hair, which has been sharpened at the end into a tip of less than 0,1 µm diameter. These elastic probe hairs allow us to establish electrical contact with on-chip bus lines without damaging them. We connect them 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 the pins of the test package. Some testing laboratories have sets of nine microprobes so that the card bus can be read out during real time operation[44].

Just to give a order-of-magnitude idea of cost ranges, complete microprobing workstations cost tens of thousands of euro, with the more luxurious versions reaching well over a hundred thousand euro. The cost of a new laser cutter is roughly in the same region. Low-budget attackers are likely to get a cheaper solution on the second-hand market for semiconductor test equipment. With patience and skill it should not be too difficult to assemble all the required tools for even under ten thousand euro by buying a second-hand microscope and using self-designed micropositioners. The laser is not essential for first results, because vibrations in the probing needle can also be used to break holes into the passivation.

Microprobing attacks can be further extended by the usage of a FIB, as said before, which can be used not just to explore the silicon, but even to modify the chip structure by creating new interconnect lines and even new transistors. Let's take, for instance, the Mondex[45] project. A TNO team lead by Ernst Bovenlander broke the smart card chipset used in this infamous "electronic purse" scheme (a 3101 controller) by microprobing. They simply fused a link in the card processor, which while intact activates a test mode in which all card contents are simply dumped to the serial port. Tom Rowley of National Semiconductor reported a similar attack on an unnamed chip using an ion beam to rewrite the link.

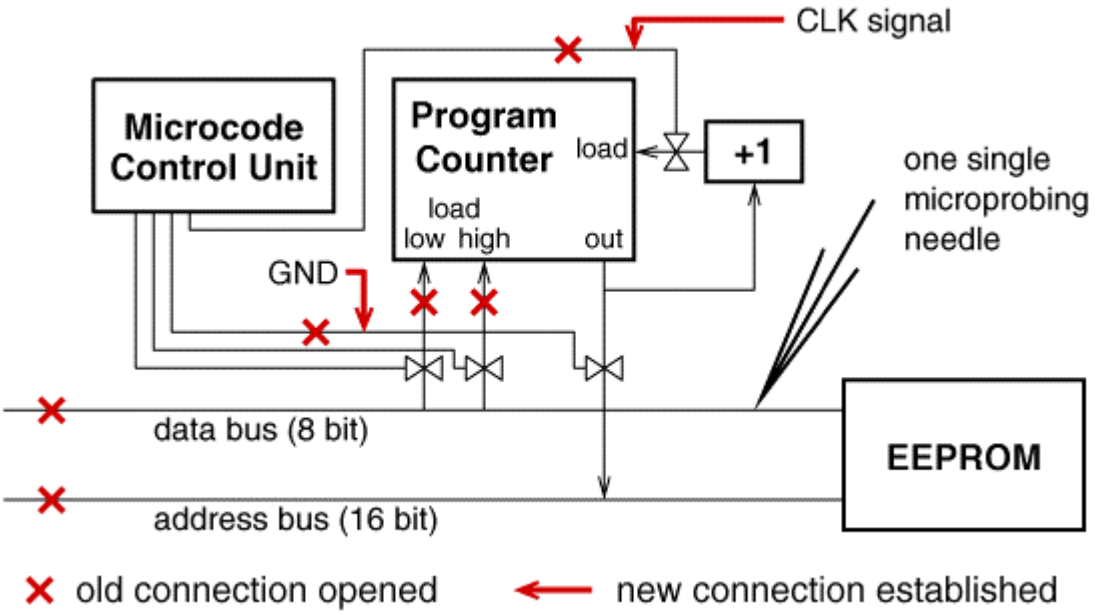
These weaknesses were reputedly fixed in the 3109 chip, by reducing the scale of chip technology from 1.3 microns (in the 3101) to 0.8 microns which substantially increases the difficulty of conventional physical probing or memory imaging type attacks, However, simply using a FIB to plate a nice large contact for the microprobe on each bus line is a widely known solution for eliminating the 0.8 micron difficulty. As always, the safest thing to do is to change the physical architecture of the chip to eliminate completely the "test mode memory access links".

In addition, with 2 microprobe needles any given bit in an EEPROM can be set or reset[46] (this makes it trivial to extract a key which parity is a precondition for the algorithm, for instance), and with a laser cutter microscope any given bit in a ROM can be modified as well. Another impressive attack by Biham and Shamir against DES hardware implementation was presented at the 1997

workshop on Fast Software Encryption, and uses a laser cutter to destroy an individual gate in the hardware implementation of a known block cipher, namely the last bit of the carry register that feeds the output of a round as the input of the next. 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 (see below), and enough of the key can be recovered to make keysearch easy.

This is an extremely impressive attack, and in fact the first one that works against ciphers such as DES when the plaintext is completely unknown. This is the case in many smartcard applications where the card uses successive transactions to report its internal state to the issuer. There is a simple countermeasure to this new attack[69]: a chip modified in this way will have the property that encryption and decryption are no longer inverses. So a simple self-test procedure can be added that takes an arbitrary input, encrypts and decrypts under an arbitrary key, and compares the result with the original block.

Another, rather typical attack involves disconnecting almost all of the CPU from the bus, leaving only the EEPROM and a CPU component that can generate read accesses. In order to read out all memory cells without the help of the card software, we have to abuse a CPU component as an address counter to access all memory cells for us. 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 us as an address sequence generator[60]. We only have 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 right metal interconnect with a laser, often have the desired effect.



Once this has been done, the attacker needs only a single microprobing needle or electro-optical probe to read the entire EEPROM contents. This makes the program much easier to analyse than in passive attacks, which typically yield only an execution trace; it also avoids the considerable

mechanical difficulties of keeping several probes simultaneously located on bus lines that are perhaps a micrometre wide.

For some chipsets this does not work. For example some banking cards read critical data from memory only after authenticating somehow that they are indeed talking to an authorized CAD. In addition, we could in theory add separate watchdog counters that reset the processor if no jump, call, or return instruction is executed for an amount of time. However such devices can be disabled by slight modification of the chipset, so it's far better to embed the protection in the structure of the chipset itself, for instance by not providing at all a program counter that can run over the entire address space. A 16-bit program counter can easily be replaced with the combination of a say 7-bit offset counter O and a 16-bit segment register S, such that the accessed address is S + O. Instead of overflowing, the offset counter resets the processor after reaching its maximum value. Every jump, call, or return instruction writes the destination address into S and resets O to zero. The processor will now be completely unable to execute more than 127 bytes of machine code without a jump, and no simple FIB edit will change this. A simple machine-code postprocessor must be used by the programmer to insert jumps to the next address wherever unconditional branches are more than 127 bytes apart[47].

With the program counter now being unavailable, attackers will next try to increase the number of iterations in software loops that read data arrays from memory to get access to all bytes. This can for instance be achieved with a microprobe that performs a glitch attack directly on a bus-line. Programmers who want to use 16-bit counters in loops should keep this in mind.

As a first line of defense against this kind of attacks, most smartcard operating systems write sensitive data to the EEPROM area in a proprietary, encrypted manner so that it is difficult to obtain cleartext keys by directly hacking into the EEPROM. This is by no means a solution (unless encryption is somehow entwined with a secret, i.e. a user supplied PIN), but everything that throws difficulty at the attacker can of course help somehow.

Additionally[47], metallization layers that form a sensor mesh above the actual circuit and that do not carry any critical signals remain one of the more effective annoyances to physical attacker. They are found in a few smartcard CPUs such as the ST16SF48A or in some battery-buffered SRAM security processors such as the DS5002FPM and DS1954.

A sensor mesh in which all paths are continuously monitored for interruptions and short-circuits while power is available prevents laser cutter or selective etching access to the bus lines. Mesh alarms should immediately set to zero all the non-volatile memory for all plaintext cryptographic keys and other unprotected critical security parameters. A well-designed mesh can make attacks by manual micro probing alone rather difficult, and more sophisticated FIB editing procedures will be required to bypass it. This is all fine in theory, but real implementations seldom keep up with this quality expectancies. In most implementation, a mesh breach will merely raise a flag, expecting card software to take necessary countermeasures. Since we know that in most probing attacks the instruction decoder is severely tampered, we cannot trust it to take any action correctly.

There's an additional danger to avoid. Micro controller production has a yield of typically around 95%, so each has to be thoroughly tested after production. Test engineers, like microprobing attackers, have to get full access to a complex circuit with a small number of probing needles. On normal processors, the test circuitry is left fully intact after the test. In smart card processors, it is common practice to blow polysilicon fuses that disable access to these test circuits. However, attackers have been able to reconnect these with microprobes or FIB editing, and then simply used

the test circuitry to dump the entire memory content. Therefore, it is essential that any test circuitry is not only slightly disabled but also structurally destroyed by the manufacturer.

One approach is to place the test interface for chip n onto the area of chip n + 1 on the wafer, such that cutting the wafer into dies severs all its parallel connections. A wafer saw usually removes a 80-200 µm wide area that often only contains a few process control transistors. Locating essential parts of the test logic in these cut areas would eliminate any possibility that even substantial FIB edits could reactivate it.

There are also other hidden dangers; for instance sometimes you may 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 gives the attacker of course easy immediate access to all memory locations on the bus and simplifies completing the read-out operation considerably. Surprisingly, such memory integrity checks were even suggested in the smartcard security literature[48], in order to defeat a proposed memory rewrite attack technique[69]. This demonstrates the importance of training the designers of security processors and applications in performing a wide range of attacks before they start to design countermeasures. Otherwise, measures against one attack can far too easily backfire and simplify other approaches in unexpected ways.

3.2.5 SEMI-INVASIVE ATTACKS

A new category of attacks has been proposed by Sergei Skorobogatov[49], that shares a part of the characteristics of invasive attacks. Like invasive attacks they require depackaging the chip in order to get access to the chip surface. However, the passivation layer of the chip remains virgin, as semi-invasive methods do not require depassivation or creating contacts to the internal lines, since microprobing is not used for this attack technology.

Semi-invasive attacks could be performed using such tools as UV light, X-rays and other sources of ionizing radiation, lasers and electromagnetic fields. They can be used individually or in conjunction with each other. For instance, we have found on the Internet a simple attack on the COP8782[50] to select carefully the area of the EEPROM containing the security bit to clear it.

Once the layout and function of the chip are known, there is an extremely powerful technique developed by IBM for observing it in operation, without having to remove the passivation layer. The tester places a crystal of lithium niobate over the feature whose voltage is to be monitored. The refractive index of this substance varies with the applied electric field, and the potential of the underlying silicon can be read out using an ultraviolet laser beam passed through the crystal at grazing incidence. The sensitivity of this technique is such that a 5 V signal of up to 25 MHz can be read[51], and we understand that it is a standard way for well funded laboratories to recover crypto keys from chips of known layout. When attacking a smartcard, for example, we would read the EEPROM output amplifiers.

Another attack can be conducted simply by illumination[52], since illumination of a target transistor causes it to conduct, thereby inducing a transient fault. Such attacks are very cheap to mount, and are finely grained and powerful: it has been demonstrated that it is possible to could change any individual bit of an SRAM array. This can be used to implement the attack of Boneh on RSA signatures against at least one smartcard currently on the market. Further details were not disclosed since the author was fearful of prosecution under the DMCA. A solution suggested by the author is the use of self-timed dual-rail logic. You can check below at paragraph 3.3.6 dangers and solutions for this kind of attack.

Comparing with non-invasive attacks, semi-invasive attacks are harder to implement as they require depackaging of the chip. However, sometimes very much less expensive equipment is needed than for invasive attacks. And these attacks can be performed in a reasonably short time. There's still a lot of research to do on this topic.

3.2.6 DENYING WRITE ACCESS

Designers of EEPROM based devices face a problem: erasing the charge stored in the oating gate of a memory cell requires a relatively high voltage. If the attacker can somehow avert this, then the information will be "unerasable".

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

Some cards are still vulnerable to this kind of attack, and it gives rise to a sporadic failure mode of some card-based public telephone systems: telephones where the relevant contact is dirty or bent may fail to decrement any user's card. However, the cards used nowadays in pay-TV decoders generate the required 12 V from the normal 5 V power supply using an on-chip oscillator and diode/capacitor network.

This pushes up the cost of an attack, but does not make it impossible: large capacitors can be identified under a microscope and destroyed with lasers, ultrasonics or FIBs.

Is this kind of failures only dangerous for prepaid phone cards or TV services ? No. As it is plainly obvious, a chip prepared in this way can be investigated at will without the risk of erasing the EEPROM, either by chance or because of some built-in protection feature.

3.3 NON INVASIVE TECHNIQUES

3.3.1 GENERALITIES

A non invasive attack on a smart card device is more limited in many ways, but has a definite advantage: since, by definition, it leaves the device completely unharmed, it can be very difficult to discover. If we could retrieve the private key of a signature device without knowledge of the user, we could forge documents in his name even for a long time before our cheating is discovered.

Sometimes non-invasive attacks can even be performed in a small amount of time, and with usual hardware (maybe with a bit of hacking). But these techniques also face hard limitations: since they must occur while a card is still operating in a black box fashion, any manipulation must be performed on the bytes entering and exiting the smartcard, or on the environmental conditions. In general a non-invasive attack requires that the software and hardware of the smart-card are known to the attacker.

3.3.2 TIMING ATTACKS

One example is the so-called "timing attack" described by Paul Kocher[53]. In this attack, various byte patterns are sent to the card to be signed by the private key. Information such as the time required to perform the operation and the number of zeroes and ones in the input bytes are used to eventually obtain the private key. There are logical countermeasures to this attack but not all smartcard manufacturers have implemented them. This attack requires that the attacker knows the the PIN to the card, or can trick the user into signing the byte patterns of his choosing (it is a chosen-plaintext attack, in cryptographic terms).

3.3.3 SOFTWARE ATTACKS

A number of attacks can be performed via software. For example, a trojan horse application could be used. The rogue application must be planted on an unsuspecting user's workstation (if this seems difficult to you, remember how many users were affected by auto-executing worms employing the Outlook autoexecution bugs).

The Trojan horse waits until the user submits a valid PIN from a trusted application, thus enabling usage of the private key, and then asks the smartcard to digitally sign some rogue data (for example, a legally binding contract – if the smart card is used for law-strong digital signature). The operation completes but the user never knows that their private key was just used against their will.

The countermeasure to prevent this attack is to use a "single-access device driver" architecture. With this type of architecture, the operating system enforces that only one application can have access to the serial device (and thus the smartcard) at any given time. This prevents the attack but also lessens the convenience of the smartcard because multiple applications can not use the services of the card at the same time.

Another way to prevent the attack is by using a smartcard that enforces a "one private key usage per PIN entry" policy model. In this model, the user must enter their PIN every single time the private key is to be used and therefore the Trojan horse would not have access to the key. This is also rarely convenient for the end user experience.

3.3.4 POWER ANALYSIS

Another possible attack is power analysis, in which we measure the fluctuations in the current consumed by the device. The various instructions cause different levels of activity in the instruction decoder and arithmetic units; they can often be quite clearly distinguished, and parts of algorithms can be reconstructed.

These techniques fall into the category of information monitoring and they are of great concern because a very large number of vulnerable products are on the market today. The attacks are easy to implement, can be automated (and so can be used by low-skill attackers) have a very low cost per device, and are non-invasive. A computer will locate correlated regions in a device's power consumption.

Simple Power Analysis involves directly observing a system's power consumption to obtain informations on the sequence of instructions executed. If the attacker has access to just 1 transaction, a limited number of informations can be leaked. Attackers with access to multiple transactions, and with knowledge of the internal mechanisms of the particular chipset in use, can be more and more challenging. Kocher[54], along with a complete description of this technique, explains how DES can be broken with it, if poorly implemented in hardware, and how this attack can be

easily averted by avoiding certain conditions, i.e. by avoiding that key material is used to choose between two branches of a jump.

DPA is based on the phenomenon that storing a 1-bit in a flip-flop consumes typically more power than a 0-bit. Also, state changes typically cause extra power consumption. 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, but there are good techniques for treating such problems[55]. In such cases, it is still often possible to compromise the system using statistical functions tailored to the target algorithm.

Using a 10-15 ohm resistor in the power supply, we can measure with an analog/digital converter the fluctuations in the current consumed by the card. Preferably, the recording should be made with at least 12-bit resolution and the sampling frequency should be an integer multiple of the card clock frequency. Drivers on the address and data bus often consist of up to a dozen parallel inverters per bit, each driving a large capacitive load. They cause a significant power-supply short circuit during any transition. Changing a single bus line from 0 to 1 or vice versa can contribute in the order of 0.5-1 mA to the total current at the right time after the clock edge, such that a 12-bit ADC is sufficient to estimate the number of bus bits that change at a time.

SRAM write operations often generate the strongest signals (a short circuit, in practice). By averaging the current measurements of many repeated identical transactions, we can even identify smaller signals that are not transmitted over the bus. 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 microcode execution, which then cause a clear change in the power consumption.

The various instructions cause different levels of activity in the instruction decoder and arithmetic units and can often be quite clearly distinguished, such that parts of algorithms can be reconstructed. Various units of the processor have their switching transients at different times relative to the clock edges and can be separated in high-frequency measurements.

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[54]. The same study shows that it is possible to reverse-engineer even unknown algorithms and protocols. In general, signals leaking during asymmetric operations tend to be much stronger than those from many symmetric algorithms, for example because of the relatively high computational complexity of multiplication operations. As a result, implementing effective SPA or DPA countermeasures can be challenging.

DPA can be used to break implementations of almost any symmetric or asymmetric algorithm. For example, a 128-bit Twofish secret key, which is considered to be safe, was recovered from a smart card after observing 100 independent encryptions[56]. In this case we can easily see that DPA reveals 1 to 2 bits of information per encryption.

The only reliable solution to DPA involves designing cryptosystems with realistic assumptions about the underlying hardware. However, there are techniques for preventing DPA and related attacks. These attacks fall roughly into three categories.

Firstly we can reduce signal size, such as by using constant execution path code, choosing operations that leak less information in their power consumption or adding extra gates to compensate for the power consumption.Unfortunately such signal size reduction cannot reduce the signal size to zero and an attacker with an infinite number of samples will still be able to perform DPA on the signal.

Secondly we may introduce noise into power consumption measurements but like in the previous case, an infinite number of samples will still enable statistical analysis. In addition, execution timing and order can be randomized. Designers and reviewers must approach temporal obfuscation with great caution because many techniques can be used to bypass or compensate for these effects.

A final approach involves using non-linear key update procedures. For example, hashing a 160-bit key with SHA should effectively lose all partial information an attacker might have gathered about the key. Similarly, aggressive use of exponent and modulus modification processes in public key schemes can be used to prevent attackers from gathering data across large numbers of operations. Key use counters can prevent attackers from obtaining large numbers of samples.

3.3.5 ELECTROMAGNETIC ANALYSIS

An attack with strong similarities with DPA is EMA, ElectroMagnetic Analysis. The idea of this attack is to measure the field radiated by the processor and correlate it to the activities of the processor. An article[57] shows that the electromagnetic attack obtains at least the same result as power consumption and consequently must be carefuly taken into account.

3.3.6 FAULT GENERATION ATTACKS

Fault generation attacks rely on stressing a card processor in order to make it perform illegal operations or give faulty results. There is a wild variety of forms these attacks can assume.

Sometimes, we will be playing around with the supply voltage and clock signal. Under-voltage and over-voltage attacks can be used to disable protection circuits or force processors to do the wrong operations. Power and clock transients can also be used to affect the decoding and execution of individual instructions. By varying the parameters, the CPU can be made to execute a number of completely different wrong instructions. Sometimes it can be fairly simple to conduct a systematic search.

Some examples: for the PIC16C84[58] microcontroller, a trick has become widely known that involves raising VCC to VPP during repeated write accesses to the security bit. This can often clear it without erasing the remaining memory.

For the DS5000[59] security processor, a short voltage drop sometimes released the security lock without erasing secret data. Processors like the 8752 that can be used with both internal and external memory but that limit the switch between them to resets have been read out using low voltages to toggle the mode without a reset. Low voltage can facilitate other attacks too: at least one card has an on-board analogue random number generator, used to manufacture cryptographic keys and nonces, which will produce an output of almost all 1's when the supply voltage is lowered slightly.

Other times, normal physical conditions, such as temperature, are altered in order to gain access to sensitive information on the smartcard. Other physical attacks that have proven to be successful involve an intense physical fluctuation at the precise time and location where the PIN verification takes place. Thus, sensitive card functions can be performed even though the PIN is unknown. This

type of attack can be combined with the logical attack mentioned above in order to gain knowledge of the private key.

Power and clock transients can be used in some processors to affect the decoding and execution of individual instructions. In a glitch attack, we deliberately generate a malfunction that causes one or more transistors to adopt the wrong state.

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

So if we apply a clock glitch (a clock pulse much shorter than normal) or a power glitch (a rapid transient in supply voltage), this will affect only some transistors in the chip. By varying the parameters, the CPU can be made to execute a number of completely different wrong instructions, sometimes including instructions that are not even supported by the microcode. Glitches can also aim to corrupt data values as they are transferred between registers and memory.

Although we do not know in advance which glitch will cause wrong instructions in a specific chip, it can be found relatively easy by a systematic search[60]. Glitch attacks seem to be most useful in practical attacks[47].

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

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

We can look for a glitch that increases the program counter as usual but transforms either the conditional jump in line 3 or the loop variable decrement in line 6 into something else. Finding the right glitch means operating the card in a repeatable way. All signals sent to it have to arrive at exactly the same time after reset for every test run. Many glitches can be tested for every clock cycle, until one of them causes an extra byte to be sent to the serial port. Repeating it causes the loop to dump the remaining memory, which if we are lucky will include the keys we are looking for.

Conditional jumps create windows of vulnerability in the processing stages of many security applications that often allows us to bypass sophisticated cryptographic barriers by simply preventing the execution of the code that an authentication attempt was unsuccessful. Output loops are just one target for glitch attacks. Others are checks of passwords, access rights and protocol responses, where corruption of a single instruction can defeat the protection.

A possible software countermeasure might be to avoid single-point-of failure instructions. This was common enough in the old days of unreliable hardware: a senior Cambridge computer scientist

recalls that in the 1950's a prudent system programmer was someone who, having masked off three bits, would verify that the result did not exceed seven!

Hardware countermeasures include independent internal clock generators that are only PLLs synchronized with the external reference frequency, but this solution has several disadvantages: PLLs are large, require a lot of time to stabilize, and are extremely sensitive to external conditions. A more extreme, but potentially more beneficial solution would be to eliminate completely the clock, transforming card processors in self timed asynchronous circuits[61]. Then, the external clock will be used as a reference only for communication: thus, clock glitches will just cause data corruption. As an additional benefit, small bit-wise or nibble-wise self-timed circuits operating at a high rate can easily outperform slowly clocked parallel circuits, while reducing silicon area. At the same time, asynchronous internal operation makes analysis of the power signature difficult, since dual-rail encoded data drastically reduces data dependent power consumption, removing any useful power signature. Self-timed circuits are even less susceptible to power glitches.

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

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

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

3.3.7 DIFFERENTIAL FAULT ANALYSIS

Differential fault analysis (DFA) is a powerful attack on crypto systems embodied in devices such as smart cards, announced by Eli Biham and Adi Shamir in a paper[62]. 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.

DFA breaks DES with 200 cyphertexts in which one-bit errors have been introduced[63]. All previous attacks required trillions. The fault model they used had been proposed by Boneh and others[64] and its effects investigated further[65 66]. It assumes that by exposing a processor to a low level of ionising radiation, or some other comparable insult, that one-bit errors can be induced in the data used and

specifically in the key material fed into the successive rounds. In addition, it has been shown[67] that this method could be extended to reverse engineer algorithms whose structure is unknown. In each case, the critical observation is that errors that occur in the last few rounds of the cipher leak information about the key, or algorithm structure, respectively. In Boneh's work it is shown that a similar fault model could be indeed applied to public key systems, by factoring an RSA modulus with a given number of faulty signatures.

There is still discussion, however, on the true generality of this attack. Paillier, in a recent paper[68], shows that under slightly modified assumptions, the DFA could be made not polynomial and would simply result in the loss of some key-bits, and proves the existence of cryptosystems on which DFA cannot reach the workfactor announced by Biham and Shamir. Other criticism was drawn by the fact that in 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 behaviour would be more likely to crash the processor or yield an uninformative error than to produce a faulty ciphertext of the kind required for the above attacks.

However, Anderson and Kuhn demonstrated[69] the ability to attack RSA or DSA by inducing faults not in the key material, but in the algorithm, by using the glitch technique we explained earlier. In particular the so-called Lenstra attack on RSA is interesting. If a smartcard computes an RSA signature, S, on a message M, modulo n = p*q by computing it modulo p and q separately and then combining them using the Chinese Remainder Theorem, and if an error can be induced in either of the former computations, then we can factor n at once. If e is the public exponent, and the signature $S = M^d \pmod{pq}$ is correct modulo p but incorrect modulo q, then we will have $p = \gcd(n; S^e - M)$

This is ideal for a glitch attack. As 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, we do not have to be selective about where in the instruction sequence the glitch is applied. Since only a single signature is needed, the attackcan be performed online (while performing a transaction for example !).

DES attacks are also pretty straightforward, if attacker can choose which instructions will fail after a glitch. For example he could remove 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 he would 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, the attacker could obtain about five bits of information about the eight keybits that were not XOR'ed as a result of the induced fault. So, for example, six ciphertexts with faulty last rounds could give away about 30 bits of the key, leaving an easy keysearch.

An even faster attack is to reduce the number of rounds in DES to one or two by corrupting the appropriate loop variable or conditional jump, as in the protocol attack described above. Then the key can be found by inspection. The practicality of this attack depends on the implementation detail.

Thus DES can be attacked with somewhere between one and ten faulty ciphertexts, if we assume that we will be able to target a particular instruction. Is that realistic ? In most smartcards, the manufacturer supplies a number of routines in ROM. Though sometimes presented as an `operating system', the ROM code is more of a library or toolkit that enables application developers to manage communications and other facilities. Its routines usually include the DES algorithm (or a proprietary algorithm such as Telepass), and by buying the manufacturer's smartcard development toolkit (for

typically a few thousand dollars) an attacker can get full documentation plus real specimens for testing. In this case, individual DES instructions can be targeted.

When confronted with an unfamiliar implementation, we may have to experiment somewhat (we have to do this anyway with each card in order to find the correct glitch parameters). However the search space is relatively small, and on looking at a few DES implementations it becomes clear that we can usually recognise the effects of removing a single instruction from either of the last two rounds. (In fact, many of these instructions yield almost as much information when removed from the implementation as the key XOR instructions do). We can also apply clock and power glitches until simple statistical tests suddenly show a high dependency between the input and output bits of the encryption function, indicating that we have succeeded in reducing the number of rounds.

Even if facing an unknown block cipher DFA can be of use. Biham and Shamir show that under their assumption the structure of an algorithm such as DES can be sketched out with about 500 ciphertext, while with 10.000 ciphertexts the complete S-boxes and all details of DES can be reconstructed. Anderson and Kuhn demonstrated that instruction glitches are even more effective in reconstructing algorithms. As we said before, secrecy should never reside in algorithms; however, if it is the case that an algorithm must be kept secret, it should be designed to be complex to deconstruct, with large S-boxes kept in EEPROM nd designing the key schedule so that the keys from a small number of rounds are not enough for a break.

The obvious defense against DFA, however, is to add error-checking to the cryptographic device. If it never delivers erroneous output, DFA becomes impossible. If erroneous output is made rare enough, DFA becomes impractical. There are a number of papers proposing error checking techniques[70]. We will just sum them up.

A first, obvious but naïve protection would be to perform the encryption twice (or more times), compare the results, and reject them if they were not equal (or implement a voting scheme). The goal is to find methods more efficient than that. Two general mechanism that have been proposed are parity checks and modular arithmetic checks. These are powerful, but not sufficient, since they can protect many, but not all cryptographic operations. Let’s see briefly why, beginning with parity.

The bit parity of a collection of bits does not change if the bits are permuted. Parity-checking, then, can protect any cryptographic operation that amounts to a permutation of bits. This covers several operations which are subsets of full bitwise permutations. Byte or block permutations, circular shifts, key scheduling for IDEA[71] or for at least some versions of CAST. It also covers all operations equivalent to the null permutation, storage & retrieval or transmission & reception of some value.

Concatenate two or more collections of bits into a larger collection and the bit parity of the result is the XOR of the parities of the inputs. Parity-checking, then, can protect any cryptographic operation that consists of concatenating bits, or the inverse, splitting a large object up into parts.

Parity-checking can also protect the XOR operation which is so common in cryptography since the bit parity of the result of an XOR is the XOR of the parities of the inputs.

What about arithmetic checking ? Any arithmetic operation on integers can be checked by performing the same operation modulo any convenient base. Using a larger base, or checking against more than one base improves the odds of catching an error. Schoolchildren check their arithmetic by "casting out nines", repeatedly adding the digits of their operands to reduce them to

values mod 9. They could get a stronger check by casting out 99s, adding pairs of digits to get values mod 99. Or trios for mod 999 or.... A similar technique works on computers, adding up groups of n binary digits to get values mod 2^n-1.

Break a number up into 3-bit groups, sum the groups (repeating if necessary) and you get its value mod 7. Use 5-bit groups, get mod 31. Do both and you're effectively checking mod 217 since 7*31=217.

There are, however, complications: for example, if you're checking an addition of two 32-bit numbers, you must check against a 33-bit result. The result checked must include the carry bit or the modular check won't work correctly. Once you've done the check, you can discard the carry bit if that's what your algorithm requires, for example if it uses arithmetic mod 2^32.

Similarly, checking a multiplication of two 32-bit numbers requires a 64-bit result as input to the modular check calculation. Checking a 32-bit division or modulo calculation requires two 32-bit results, quotient & remainder. This might be a major problem in the constrained environment of smartcard hardware design. Checking a calculation such as IDEA's multiplication mod 2^16+1 is certainly possible, and may not even be problematic if your smart card has a 32-bit CPU in it, but it is far from clear that it will be practical in all situations.

Let's see examples of applications of this technique, for instance an application to Linear Feedback Shift Registers. LFSRs are a staple of stream cipher design. They can fairly easily be protected against DFA with parity-check logic plus a flip-flop. Consider an LFSR of the configuration which XORs some Boolean constant into the register whenever the output bit is one. When a zero is output, the parity of the register does not change.

When a one is output, that changes the parity. If the XOR changes the parity back, then overall the parity does not change. The parity of the register is then constant. It cannot possibly be a maximal period LFSR since it does not cycle through all values, only (at most) the half of them with that parity.

For the LFSR to have maximum period, the parity must sometimes change. It clearly does not change on zero output & if it sometimes changes on one, it must always do so since the XOR is with a constant. So the parity of the shift register contents of a maximal period LFSR changes whenever a one bit is output. Initialize a flip-flop with the initial parity of the shift register. Flip it every time a one is output. Check it against the register parity periodically. This protects any maximal period LFSR against errors at very low cost.

Let's see how we can protect the substitution tables or S-boxes that are used in many ciphers (DES, CAST…). Protecting them requires an extended version of parity checking. Consider the F function in the second version of CAST. All of this cipher, except the F-function and key scheduling are easily protected by parity checks, since they only use XORs and swaps.

The F function inputs are a 32-bit round key & 32-bit text block. The output is a 32-bit result. XOR key and text, split the 32-bit result into four bytes, pass each through a different S-box, and XOR the four 32-bit S-box outputs to get the 32-bit result. To protect this, extend each S-box row from 32 to 34 bits. At S-box setup time, calculate the parity of each 32-bit S-box row & store that in the 33rd bit of the row. Also find the parity of the corresponding S-box input byte & store that in the 34th bit of the row.

The encryption is exactly as described above except that the S-box rows, the four-way XOR & the its result are all 34-bit. Check the 33rd bit of that result against the parity of the 1st 32 bits. This checks the read of S-box memory & the XORs.

Check the 34th bit against the parity of the two inputs, the round key & the text (concatenated or XORd together). This should match since the 34th result bit is the XOR of the four S-box 34th bits which are the parities of the S-box input bytes which are the four bytes of the XORd key & text.

These two bits together, then, protect the entire F function. Add parity checks outside the function and we can protect all of CAST against DFA reasonably cheaply.

CAST is not the only cipher that can be entirely protected with techniques shown here. GOST is one other.

3.3.8 COMMON COUNTERMEASURES AGAINST NON INVASIVE TECHNIQUES

Many non-invasive attacks require the attacker to predict the time at which a certain instruction is executed. A strictly deterministic processor that executes the same instruction n clock cycles after each reset, if provided with the same input at every cycle, makes this easy.

The obvious countermeasure is to insert random time delays between any observable reaction and critical operations that might be subject to an attack. If the serial port were the only observable channel, then a few random delay routine calls controlled by a hardware noise source would seem sufficient. However, since attackers can use cross-correlation techniques to determine in real-time from the current fluctuations the currently executed instruction sequence, a few localized delays will not suffice.

A study[47] therefore strongly recommends introducing timing randomness at the clock-cycle level. A random bit-sequence generator that is operated with the external clock signal should be used to generate an internal clock signal.

To introduce even more non-determinism into the execution of algorithms, it is conceivable to design a multi-threaded processor architecture[72] that schedules the processor by hardware between two or more threads of execution randomly at a per-instruction level. Such a processor would have multiple copies of all registers, and the combinatorial logic would be used in a randomly alternating way to progress the execution state of the threads represented by these respective register sets.

3.4 OTHER DANGERS

3.4.1 DATA REMANENCE

Smart cards, as all magnetic and electronic media, suffer from data remanence problems. Recent research[73] have shown various dangers in the usage of EEPROM and Flash memories. While leaving the details to the referred paper, we will quote some of the conclusions and design suggestions. First of all, cryptovariables should not be stored in RAM for long periods of time (neither on the smart card, nor on the PC host system). Move them around from time to time.

EEPROM/Flash cells should be “cycled” 10-100 times with random data before writing anything sensitive to them. The use of fresh cells creates biases that can be used to detect the first value written to a cell long after its deletion.

Researches made on crypto hardware (such as RSA accelerators) on hot carrier and electromigration effects make us wonder about any possible residuates of data remaining on the crypto coprocessor of cards, but as of the time of writing there are no researches we are aware of on this subject.

If the flash RAM devices are “intelligent” (i.e. they use wear-levelling techniques to try to optimize the usage of Flash, or to avoid unnecessary and time consuming block-erase commands) they may leave copies of your sensitive data around in mapped-out memory, even long after you think you erased or changed that. This is very worrying, since if they are properly implemented they won’t let you override this behaviour.

As a final note remember that the ever increasing density of semiconductor memory and the usage of exotic techniques such as multilevel storage is one of the best defenses available to make it increasingly expensive and difficult to recover data from devices. In addition all these techniques should be counted as invasive attacks, since they usually require destruction of the token.

This danger should not be underrated. There’s an astounding report[69] about a security module used in a bank. Many banks use a system devised by IBM and refined by VISA to manage the personal identication numbers (PINs) issued to customers for use with automatic teller machines[74]. The PIN is derived from the account number by encrypting it with a “PIN key”, decimalising the result and adding a decimal “offset” (without carry) to get the PIN the customer must enter. (The offset's function is to enable the customer to choose his own PIN).

The function of the security module is to perform all these cryptographic operations, plus associated key management routines, in trusted hardware, so as to support a dual control policy: no single member of any bank's staff should have access to a customer PIN[75]. Thus, for example, the module will only perform a “verify PIN” command if the PIN is supplied encrypted under a key allocated to an automatic teller machine or to a corresponding bank. In this way, bank programmers are prevented from using the security module as an oracle to perform exhaustive PIN search.

In order to enforce this, the security module needs to be able to mark keys as belonging to a particular functionality class. It does this by encrypting them with 3DES under one of 12 pairs of DES master keys that are stored in low memory. Thus for example ATM keys are stored encrypted under master keys 14 and 15, while the working keys used to communicate with other banks are stored encrypted under master keys 6 and 7. The encrypted values of long term keys such as the PIN key are often included inline in application code and are thus well known to the bank's programming staff. So security depends on the module’s tamper resistance, and this is provided for by lid switches that cut power to the key memory when the unit is opened (it needs servicing every few years to change the battery). Master keys are loaded back afterwards in multiple components by trusted bank staff. Looking at one such device, which dated from the late 1980's, Anderson and Kuhn found that the master key values were almost intact on power-up. The number of bits in error was of the order of 5-10%. The almost-correct values had been “burned in” to the static RAM chips.

This level of memory remanence would be alarming enough. However, it has a particularly pernicious and noteworthy interaction with DES key parity in this common application. If each DES key is wrong by five bits, then the effort involved in searching for the 10 wrong bits in a double DES key might be thought to be 112-choose-10 operations. Each operation would involve (a) doing a 2-key 3DES decryption of a 56 bit PIN key whose enciphered value is, as we noted, widely known (b) in the 2- 8 of cases where this result has odd parity, enciphering an account number with this as a DES key to see if the (decimalised) result is the right PIN. The effort is about

3 times 112-choose-10 DES operations, approximately say 2^{50}. But it would probably be cheaper to do a hardware keysearch on the PIN key directly than to try to implement this more complex search in either hardware or software.

However, the bytewise nature of the DES key redundancy reduces the effort by several orders of magnitude. If no key byte has a double error, then the effort is seven tries for each even parity byte observed, or 3 times 7^{10}, or about 2^{30} which is easy. If there is one key byte with a double error, the effort is 2^{38}, giving a search of 2^{40} DES operations, totally feasible.

3.4.2 PIN GUESSING

Most cards are PIN-protected to avoid the risk that stolen or lost devices are fraudulently used. In most cases, users are allowed to choose and change the PIN protecting the device. This can lead to poor PINs (day and month of birth, for example), but has a fundamental strength: it does NOT rely on a (potentially poor) algorithm to choose numbers. As shown in research[76] this can make the traditional "three attempts before lock" paradigm inadequate. Another interesting research[77] shows how to design authentication and verification protocols that are resistant to "password guessing" attacks. Still another[69] shows how PIN assignment protocols can come under attack – or be plainly inadequate.

We would like to stress again that there's a time frame between when a smart card is lost and when the user realizes it and can safely revoke it. In this time frame, the PIN is the only barrier that prevents abuse of the device.

3.4.3 SOCIAL ENGINEERING ATTACKS

In computer security systems, this type of attack is usually the most successful, especially when the security technology is properly implemented and configured. Usually, these attacks rely on the faults in human beings. A common example of a working social engineering attack has a hacker impersonating a network service technician. The serviceman approaches a low-level employee and requests their password for network servicing purposes. With smartcards, this type of attack is a bit more difficult. Most people would not trust an impersonator wishing to have their smartcard and PIN for service purpose.

3.5 TRUE TAMPER SAFE DESIGN: THE MILITARY APPROACH

One application in which capable, motivated opponents may be assumed, and where billions of dollars are spent on thwarting them, is the security of nuclear weapons. The threat here are nothing less than rogue states fronted by terrorist commando teams, operating in cahoots with subverted military personnel – typical "class 3" opponents. The U.S.A. has led the development of a control technology, now partially shared with other nuclear and near-nuclear nations, and the following account is drawn from a previously quoted article[60].

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

President Kennedy's response was National Security Action Memo no. 160, which ordered that America's 7,000 nuclear weapons then in countries from Turkey to Germany should be got under positive control, or got out[78].

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

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

But, following the Kennedy memo, it was proposed that all nuclear bombs should be protected using code locks, and that there should be a `universal unlock' action message that only the president or his legal successors could give. How could this be securely translated to a large number of individual firing codes, each of which would enable a small batch of weapons? The problem became worse when the Carter administration's policy of `measured response' created a need for a wide variety of `selective unlock' messages, giving the president options such as enabling the use of nuclear artillery and air defence weapons against a Soviet incursion into Germany. It became worse still with concern that a Soviet decapitation strike against the U.S. national command authority might leave the arsenal intact but useless. As is now well known, the solution lies in the branch of cryptomathematics known as `secret sharing'[71], whose development it helped to inspire, and which enables weapons, commanders and options to be linked together with a complexity limited only by the available bandwidth.

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

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

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

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

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

The nuclear business also supplies the only examples known to us of tamper resistant packages designed to withstand a “class 3” opponent who can obtain unsupervised physical access. These are the missile sensors developed to verify the SALT II treaty[79] - which was never deployed - and the seismic sensor package developed for test ban treaty verification, which was. In this latter system, the seismic sensors are fitted in a steel tube and inserted into a drill hole that is backfilled with concrete. The whole assembly is so solid that the seismometers themselves can be relied upon to detect tampering events with a fairly high probability. This physical protection is reinforced by random challenge inspections.

So, if systems have to be protected against “class 3” opponents, we might hazard the following summary:

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

The above analysis convinced us that military agencies have limited confidence in the ability of tamper-resistant devices (and especially portable ones) to withstand a “class 3” opponent with unsupervised access. A senior agency official confirmed that chip contents cannot be kept from a capable motivated opponent; at most one can impose cost and delay. A similar opinion was ventured by a senior scientist at a leading chip maker.

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

4 CLOSING REMARKS

4.1 FOR FURTHER READING

In addition to the references quoted throughout the text, there are additional texts and sites that can add informations to the rather synthetical descriptions in this article:

4.1.1 SITES

http://www.cl.cam.ac.uk/~mgk25/
http://www.geocities.com/ResearchTriangle/Lab/1578/smart.htm
http://www.cl.cam.ac.uk/Research/Security/tamper/
http://www.usenix.org/events/smartcard99/

4.1.2 BOOKS

Renkl, W., Effing, W. "Smartcard Handbook", John Wiley & Sons, 1997
Hendry, M., "Smart Card Security and Applications, Second Edition", Artech House, 2001
Hassler, V., Gordeev, M., Manninger, M., Muller, C., Moore, P., "Java Card E-Payment Application Development", Artech House, 2001

4.2 ACKNOWLEDGMENTS

This work would never have been possible without the contribution of a number of people. In particular I need to thank ing. Francesca Fiorenza for providing me first-class references materials after a presentation she held at "BlackHats.it '01", and for reviewing a draft of this paper.

---

[1] ISO 7810 *"Identification cards – Physical characteristics"*
[2] Here and in the following we used as a reference *"Introduction to Smart Cards"*, http://www.sspsolutions.com/solutions/whitepapers/introduction_to_smartcards/?page=printerfriendly
[3] ISO 7811 Identification cards – Recording technique
[4] ISO 7816 *"Identification cards – Integrated circuit(s) cards with contacts"* http://www.geocities.com/ResearchTriangle/Lab/1578/iso7816.txt (parts 1-3), http://www.geocities.com/ResearchTriangle/Lab/1578/iso78164.htm (part 4).
[5] ISO/IEC 11693: 1995, Optical memory cards
[6] ISO/IEC 11694: 1995, Optical memory cards – Linear recording method
[7] http://www.opencard.org/
[8] ISO 7813: 1995, Identification cards – Financial transaction cards
[9] ISO/IEC 10373: 1993, Identification cards – Test methods
[10] GSM 11.11: 1995, European digital cellular telecommunications system
[11] EN 726-3: 1994, Identification card systems – Telecommunications integrated circuit(s) card and terminals
[12] prEN 1546: 1995, Identification card systems – Inter-sector electronic purse
[13] Documentation may be found at http://java.sun.com/products/javacard/
[14] http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/
[15] http://www.pcscworkgroup.com/
[16] http://developer.intel.com/ial/security/
[17] http://www.microsoft.com/workshop/security/default.asp
[18] http://www.mondexusa.com/
[19] http://www.visa.com/cgi-bin/vee/nt/chip/main.html?2+0
[20] http://www.protonworld.com/
[21] FIPS PUB 140-1, Security Requirements for Cryptographic Modules, *National Institute of Standards and Technology, U.S. Department of Commerce*, 11.1.1994 http://csrc.nist.gov/fips/fips1401.htm
[22] *CardTech/SecurTech 2000 Presentation*, Smart Card Security for the New Millennium, Miami Beach, 3.5.2000 http://csrc.nist.gov/cc/sc/troy-ctst2000.pdf
[23] Jøsang, A., The difficulty of standardizing smart card security evaluation, *Computer Standards and Interfaces Journal*, pp. 333-341, 17.9.1995 http://www.item.ntnu.no/~ajos/papers/cardeval.ps
[24] ISO 15408:1999, Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, Version 2.1, August 1999 http://csrc.ncsl.nist.gov/cc/ccv20/p1-v21.pdf
[25] ISO 15408:1999, Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements, Version 2.1, August 1999 http://csrc.ncsl.nist.gov/cc/ccv20/p2-v21.pdf

[26] *Smart Card Security User Group*, Smart Card Protection Profile Draft, Version 2.0, 1.5.2000 http://csrc.nist.gov/cc/sc/sclist.htm
[27] Reid, J. & Looi, M., Making Sense of Smart Card Security Certifications, *Fourth Working Conference on Smart Card Research and Advanced Applications*, 20-22.9.2000, Bristol, UK
[28] As demonstrated by the latest advertising campaign by Oracle Corporation: http://online.securityfocus.com/news/309
[29] DG Abraham, GM Dolan, GP Double, JV Stevens, ``Transaction Security System", in *IBM Systems Journal* v 30 no 2 (1991) pp 206-229
[30] B. Schneier, A. Shostack, "Breaking Up Is Hard To Do: Modeling Security Threats for Smart Cards", USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA, May 10–11, 1999
[31] F. Beck: Integrated Circuit Failure Analysis - A Guide to Preparation Techniques. John Wiley & sons, 1998
[32] N.H.E. Weste, K. Eshraghian: Principles of CMOS VLSI Design. Addison-Wesley, 1993.
[33] S.-M. Kang, Y. Leblebici: CMOS Digital Integrated Circuits: Analysis and Design. McGraw-Hill, 1996.
[34] J. Carter: Microprocessor Architecture and Microprogramming - A State-Machine Approach, Prentice-Hall, 1996.
[35] S.M. Sze: Semiconductor Devices - Physics and Technology. John Wiley & Sons, 1985
[36] O. Kommerling and M. G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. In USENIX Workshop on Smartcard Technology, 1999. http://citeseer.nj.nec.com/kommerling99design.html
[37] J.H. Daniel, D.F. Moore, J.F. Walker: Focused Ion Beams for Microfabrication. Engineering Science and Education Journal, pp. 53-56, April 1998.
[38] H. P. Feuerbaum: Electron Beam Testing: Methods and Applications. Scanning, 5(1):14-24, 1982.
[39] S Blythe, B Fraboni, S Lall, H Ahmed, U de Riu, 'Layout Reconstruction of Complex Silicon Chips', IEEE J. of Solid-State Circuits v 28 no 2 (Feb 93) pp 138-145
[40] C. Ajluni, 'Two New Imaging Techniques Promise To Improve IC Defect Identification', Electronic Design Vol 43 No 14 (10 July 1995) pp 37-38
[41] KE Gordon, RJ Wong, 'Conducting Filament of the Programmed Metal Electrode Amorphous Silicon Antifuse', International Electron Devices Meeting, Dec 93; reprinted as pp 6-3 to 6-10, QuickLogic Data Book, 1994
[42] M Blaze, "Protocol Failure in the Escrowed Encryption Standard", in *Proceedings of the 2nd ACM Conference on Computer and Communications Security* (2-4 November 1994), ACM Press, pp 59-67 ftp://research.att.com/dist/mab/eesproto.ps
[43] See FIPS PUB 140-1 (cited above), at section 4 level 4: "Removal of the coating shall have a high probability of resulting in serious damage to the module"
[44] E Bovenlander, RL van Renesse, ``Smartcards and Biometrics: An Overview", in *Computer Fraud and Security Bulletin* (Dec 95) pp 8-12
[45] See the press release at http://www.att.com/press/0796/960718.uca.html
[46] O Kocar, "Hardwaresicherheit von Mikrochips in Chipkarten", in Datenschutz und Datensicherheit v 20 no 7 (July 96) pp 421-424
[47] Kömmerling, O. & Kuhn, M.G., Design Principles for Tamper-Resistant Smartcard Processors, *USENIX Workshop on Smartcard Technology*, Chicago, IL, 10-11.5.1999 http://www.cl.cam.ac.uk/~mgk25/sc99-tamper.pdf
[48] D.P. Maher: Fault Induction Attacks, Tamper Resistance, and Hostile Reverse Engineering in Perspective. In R. Hirschfeld (ed.): Financial Cryptography, FC '97, Proceedings, LNCS 1318, pp. 109-121, Springer-Verlag, 1997.
[49] http://www.cl.cam.ac.uk/~sps32/
[50] See description here: http://www.geocities.com/ResearchTriangle/Lab/1578/cophack.txt
[51] JM Wiesenfeld, ``Electro-optic sampling of high-speed devices and integrated circuits", in *IBM Journal of Research and Development* v 34 no 2/3 (Mar/May 1990) pp 141-161; *see also subsequent articles in the same issue*
[52] Sergei Skorobogatov, Ross Anderson, Optical Fault Induction Attacks
[53] Paul Kocher, *"Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems"*, http://www.cryptography.com/timingattack/paper.html
[54] Kocher, P. & Jaffe, J. & Jun, B., "Differential Power Analysis", *CRYPTO '99 Proceedings*, Springer–Verlag, pp. 388-397, 1999 http://www.cryptography.com/resources/whitepapers/DPA.pdf
[55] T. S. Messerges, E. A. Dabbish, R. H. Sloan, "Investigations of Power Analysis Attacks on Smartcards", USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA, May 10–11, 1999
[56] Chari, S. & Jutla, C. & Rao, J.R. & Rohatgi, P., A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards, *AES Second Candidate Conference*, Rome, Italy, 22-23.3.1999 http://csrc.nist.gov/encryption/aes/round1/conf2/papers/chari.pdf
[57] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Couter-Measures for Smard Cards. In Smart Card Programming and Security (E-smart 2001), Cannes, France, LNCS 2140, pp.200-210. September 2001.
[58] See, for instance, http://www.geocities.com/ResearchTriangle/Lab/1578/pichack.txt
[59] Kuhn, M. "Sicherheitsanalyse eines Mikroprozessors mit Busverschlüsselung", Diploma Thesis (in german). http://www.cl.cam.ac.uk/~mgk25/kuhn-da.pdf

[60] Anderson, R. & Kuhn, M.G., Tamper Resistance – a Cautionary Note, *Second USENIX Workshop on Electronic Commerce Proceedings*, pp. 1-11, Oakland, California, 18-21.11.1996 http://www.cl.cam.ac.uk/~mgk25/tamper.html
[61] Simon Moore, Ross Anderson & Markus Kuhn "Improving Smartcard Security Using Self-timed Circuit Technology", Proceedings of the Eighth International Symposium on Asynchronous Circuits and Systems.
[62] E. Biham, A. Shamir, "Differential fault analysis of secret key cryptosystems", Advances in Cryptology - Crypto '97, Lecture Notes in Computer Science, vol. 1294, Springer, Berlin, pp. 513--525, 1997. http://citeseer.nj.nec.com/biham97differential.html
[63] E Biham, A Shamir, "A New Cryptanalytic Attack on DES", preprint, 18/10/96
[64] D Boneh, RA DeMillo, RJ Lipton, "On the Importance of Checking Computations", preprint, 11/96
[65] M Joye, F Koeune, JJ Quisquater, "Further results on Chinese remaindering", Universitè Catholique de Louvain Technical Report CG-1997-1 <http://www.dice.ucl.ac.be/crypto/tech_reports/CG1997_1.ps.gz>
[66] F Bao, RH Deng, Y Han, A Jeng, AD Nirasimhalu, T Ngair, "Breaking Public Key Cryptosystems in the Presence of Transient Faults", Security Protocols, 5th International Workshop, Paris, France, April 7/9, 1997, Proceedings, Springer LNCS 1361
[67] E Biham, A Shamir, "Differential Fault Analysis: Identifying the Structure of Unknown Ciphers Sealed in Tamper-Proof Devices", preprint, 10/11/96
[68] Paillier, P. "Evaluating Differential Fault Analysis of Unknown Cryptosystems", in Public-Key Cryptography, vol. 1560 of Lecture Notes in Computer Science, pp. 235-244, Springer-Verlag, 1999. http://www.gemplus.com/smart/r_d/publi_crypto/download/346951_1290114397.pdf
[69] Anderson, R., Kuhn, M., "Low Cost Attacks on Tamper Resistant Devices", Security Protocols, 5th International Workshop, Paris, France, April 7{9, 1997, Proceedings, Springer LNCS 1361, pp 125-136
[70] For example, see: Sandy Harris, *"Defending against DFA"*, posted on 30/01/97 in sci.crypt.research
[71] Schneier, B. *"Applied Cryptography, 2nd edition"*, J. Wiley and sons, 1996
[72] S.W. Moore: Multithreaded Processor Design. Kluwer Academic Publishers, 1996.
[73] P Gutman, "Secure Deletion of Data from Magnetic and Solid-State Memory", in Sixth USENIX Security Symposium Proceedings (July 1996) pp 77-89 http://www.cryptoapps.com/~peter/usenix01.pdf,
[74] RJ Anderson, "Why Cryptosystems Fail", in Proceedings of the 1st ACM Conference on Computer and Communications Security (November 1993) pp 215-227
[75] "VISA Security Module Operations Manual", VISA, 1986
[76] Markus G. Kuhn, "Probability Theory for Pickpockets: ec-PIN Guessing"
[77] L.Gong, M.Lomas, R.Needham, J.Saltzer, "Protecting Poorly Chosen Secrets from Guessing Attacks", IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, (1993), pp. 648--656. http://citeseer.nj.nec.com/gong93protecting.html
[78] GJ Simmons, invited talk at the *1993 ACM Conference on Computer and Communications Security*, Fairfax, Virginia, Nov 3-5, 1993
[79] GJ Simmons, ``Subliminal Channels; Past and Present", *European Transactions on Telecommunications* v 5 no 4 (Jul/Aug 94) pp 459-473