LEGACY SMARTCARD'S ATTACKS

Jaume Boniquet Travila
20 May 2014, Barcelona, SPAIN
jboniquet@securityaversion.net

Abstract

Smart Cards are becoming an everyday use device. Nowadays almost everyone has a Smart Card, such as mobile SIM cards, Access Control Cards for accessing buildings, payment cards, and so on. Though the use of different technologies, protocols and communications mechanisms, most of them share a common standard. This Standard is called ISO/IEC 7816.

The protocol by itself is not vulnerable to specific attacks, but depending on its implementation can stimulate some attack vectors. This paper discusses some basic but useful attacks on such implementations, including hidden commands hardcoded and TRNG entropy[1] attacks.

1. Introduction

We usually assume that the Smart Cards are one of the most secure and reliable devices around. Almost all Smart Cards are built following the ISO/IEC 7816, which was created in 1987, updated in 1998 and amended in 2003. The author believes that this Standard is not adequate to the actual security needs for such a critical devices, as it is too vague and general purpose.

The ISO/IEC 7816 is a privative Standard. It means that is not publicly accessible, therefore a payment is required to obtain a copy of it. The Standard is hosted and supported by the "*International Organization for Standarization*"[2].

Among many other features, the ISO mandates a set of very basic security features, with optional enhanced security mechanisms.

This paper focuses in three attack vectors present on almost all Smart Cards compliant with ISO/IEC 7816.

[1] More information about entropy: http://en.wikipedia.org/wiki/Entropy_%28information_theory%29

[2] ISO (International Organization for Standardization) www.iso.org

The first attack vector consists in “bruteforcing” the internal file structure present in the Smart Card.

The second attack vector is a technique to search for hidden or undocumented APDU commands on Smart Cards devices.

The last attack vector looks for weaknesses in the randomness of the numbers generated by the crypto processors. To be specific, the entropy quality is measured. These features are only present in high end Smart Cards with extra features.

2. Related Work.

Related work to the vulnerabilities described in this paper does exist, but most of them are theory or too complex for average attackers.

A paper[3] published in the Defcon conferences suggested bruteforcing attacks on the smart cards based on FPGA’s, but no code or Proof of Concept is provided. Also this other paper [4] talks about several types of attacks in the Smart Cards world, such as Timing attacks, removing the chip from the card envelope to tamper with, reverse engineering the chip itself, power analysis attacks fault generation attacks, etc but many of them require very strong electronics skills or laboratory specific equipment to analyze the chip and lasers to produce faults.

After further research, a tool to implement the attack number two (Attacking Commands definition on Smart Cards.) does already exist. Although this tool is not updated anymore, nor does work with the latest PCSCD drivers and it does not interpret response APDU’s from the bruteforcing.

Another tool[5] for the same purpose was found, although it does only implement brutefocing over the CLA and INS attributes from an APDU command.

No tools or proof of concepts were found capable to automate the crypto analysis of True Random Data generated by Smart Cards however some previous work[6] related to attacking the RNG does exist.

3. Basic Smart Card concepts overview.

The purpose of this paper is NOT to explain how does the Smart Cards work or to be a guide explaining detailed concepts of specific features, however a basis is required to understand how the attack vectors are exploited.

[3]http://tech.mit.edu/V128/N30/subway/Defcon_Presentation.pdf
[4]http://home.deib.polimi.it/zanero/papers/scsecurity.pdf

[5]http://michau.benoit.free.fr/codes/smartcard/
[6]https://www.usenix.org/legacy/event/sec08/tech/full_papers/nohl/nohl_html/index.html

Due to the lack of working state tools to conduct a proper bruteforcing over file structure, APDU commands and random data, three scripts were developed in an easy scripting Language (the chosen was Perl) to accomplish these objectives.

3.1. File Structure.

General purpose Smart Cards compliant with ISO/IEC 7816 must implement a file structure inside the device. This file structure is somehow similar to the ones found in earlier PC Operating Systems, however, due to the device limitations, with reduced capabilities.

The file structure is composed of files, which can hold data, and folders, which can contain information files.

The most important is the MF (Master File). This can be seen as the root directory, where the headers of elementary files and dedicated files are contained.

Later on, the DF (Dedicated Files) are like ordinary folders. Very similar to the concept of folders defined in NTFS or EXT3/4 file systems.

Finally the EF (Elementary File) is the actual data container, which can store data.

The file Structure is protected by specific access conditions, which must be fulfilled before accessing the data. An example of access conditions would be an authentication against the card with a privileged user using APDU’s. This is accomplished through the header of each DF's, EF's and MF's, which contains security attributes resembling user rights associated with a file/directory in a common OS. Any application can traverse the file tree, but it can only move to a node if it has the appropriate rights

There are five basic levels of access rights to a file (both DF and EF). Some OS provide further levels. Basic levels can be categorized, increasingly in security, as follows:

1. **Always (ALW):** Access of the file can be performed without any restriction.

2. **Card holder verification 1 (CHV1):** Access can only be possible when a valid CHV1 value is presented.

3. **Card holder verification 2 (CHV2):** Access can only be possible when a valid CHV2 value is presented.

4. **Administrative (ADM):** Allocation of these levels and the respective requirements for their fulfillment are the responsibility of the appropriate administrative authority.

5. **Never (NEV):** Access of the file is forbidden.

Finally bear in mind that all file structure is saved in solid state memory, such as RAM, ROM or EEPROM memories.

To simplify the concept, a sample file structure could be as follows:

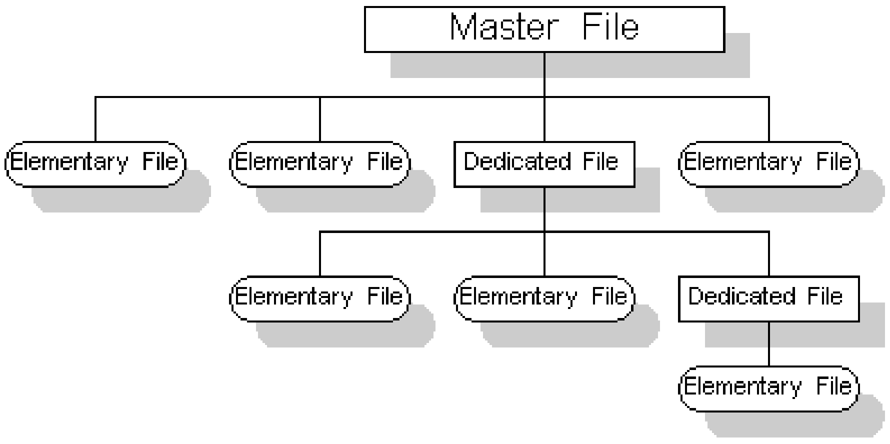


Figure 1 – File structure schema

3.2. Commands definition.

Also, Smart Cards compliant with ISO/IEC 7816 must implement a specific mechanism to communicate between the card and the reader, in order to retrieve and send data and maintain compatibility. This is defined in the *Part 4* of the Standard. This is accomplished with APDU's (*Application Protocol Data Unit*).

There are two categories of APDU. Command APDUs and response APDUs.

1. A command APDU is sent by the reader to the card (it contains a mandatory 4-byte header "CLA, INS, P1, P2" and from 0 to 255 bytes of data)
2. A response APDU is sent by the card to the reader (it contains from 0 to 65 536 bytes of data, and 2 mandatory status bytes "SW1, SW2").

The following table shows the complete parameters structure that composes an APDU. For more information about APDU construction check the ISO/IEC 7816 Part 4:

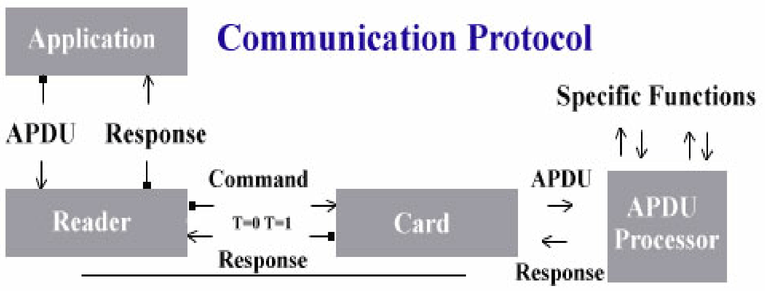
Code	Name	Length	Description
CLA	Class	1	Class of instruction
INS	Instruction	1	Instruction code
P1	Parameter 1	1	Instruction parameter 1
P2	Parameter 2	1	Instruction parameter 2
Lc field	Length	variable 1 or 3	Number of bytes present in the data field of the command
Data field	Data	variable= Lc	String of bytes sent in the data field of the command
Le field	Length	variable 1 or 3	Maximum number of bytes expected in the data field of the response to the command
SW1/SW2	Status Word	Variable	Command process status

Figure 2 – APDU construction structure

3.3. TRNG (True Random Number Generator).

Some Smart Cards include a Cryptographic processor. With such capabilities, enables smart cards to handle complex mathematical computations, such as PKI uses, personal ID purposes, digital signing, ePassports identification, etc.

The Standard ISO/IEC 7816 Part 4 does not specify how to develop a secure TRNG generator, only indicates that the seed must be secure enough. Therefore, each card manufacturer has to accomplish a minimum of requirements stated in the standard to provide compatibility between cards and readers, but the extra crypto features are optional and subject to home brewed algorithms and proprietary security measures. An example of a proprietary development card but that also shares some ISO/IEC7816 features are the Mifare[7] cards. A simplified schema of how do these kind of cards operates is shown below:



[7] http://www.mifare.net/en/home/

3.4. Encryption Algorithms used in Smart Cards.

Different Smart Cards manufacturers do design cards with support for different encryption algorithms. Cards supporting encryption features do usually implement at the same time symmetric and asymmetric algorithms. The reason for that is that very sensitive data shall not be encrypted with symmetric algorithms, but does with asymmetric algorithms such as RSA/DSA/ECC

Symmetric algorithms, such as DES/3DES, AES, are very quick to perform, as the card crypto processor has specific optimized instructions to process them. Therefore these algorithms are used to dynamically encrypt the whole card memory, file registries and attributes, and other data which is sensitive, but not critical. The following figure shows the most typical algorithms used in Smart Cards, classified by its types.

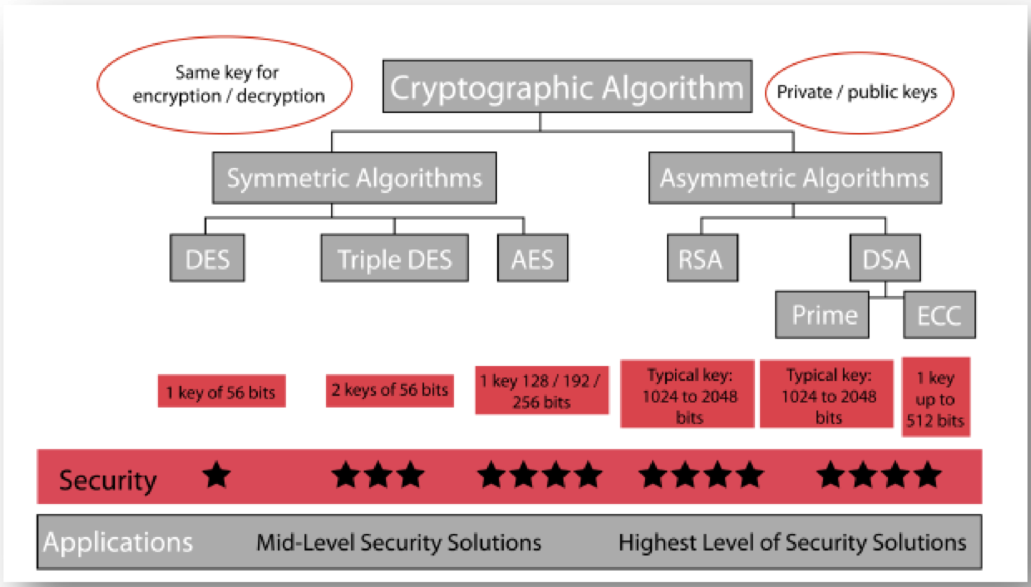


Figure 3 – Common Smart Cards Algorithms

3.5. Protection of Sensitive Data.

The main goal for a secure smart card product is to protect embedded assets and achieve the security objectives defined by the application designer. A secure product therefore has to be developed to protect sensitive data (the assets) from identified threats that could compromise any of the following:

- Confidentiality. All data classified as sensitive by the OS and application developers must be kept confidential. This protection includes, at a minimum, controlling access to IC memory.
- Integrity. The integrity of all sensitive (and any related) data or code must be controlled, including the integrity of the security functions.
- Availability. The data required by the IC must always be available, and the security functions must always be controlled and accessible.

Smart card applications vary in their requirements for data storage, processing and security.

These requirements will dictate the choice of secure IC to be used and the types of security measures that will be implemented in the application.

3.6. Types of Attackers

Attackers typically fall into one of three areas:

- <u>Amateur:</u> Amateurs are curious individuals who carry out attacks just to "see if it can be done."
- <u>Expert:</u> Experts attack under the auspices of scientific institutions and universities studying the technology.
- <u>Professional:</u> Professionals attack for financial reward or to obtain sensitive data and compromise a system.

3.7. Types of Known Attacks

Attacks are techniques implemented to compromise the security of a smart card IC by discovering what information it holds. General purpose attacks can be categorized as fault attacks, side-channel attacks, or invasive attacks.

- <u>Fault Attacks:</u> This technique consists in introducing faults to the code, trough its own features. This attacks takes advantage of unsecured methods, wrong error handling events and also comprises de Fuzzing Attacks
- <u>Side-channel attacks:</u> A side channel attack is any attack that takes advantage of the information gathered from the hardware itself. It comprises many techniques to extract useful information, such as timing

information, power consumption, electromagnetic leaks or even sound can provide an extra source of information which can be exploited to break the system.

Several papers[8],[9] have been written to exploit this attacks vectors, some theoretical, some others[10],[11] including a Proof of Concept with real exploiting results.

- Hardware attacks: These attacks are also very complex, also special hardware equipment is often required to conduct this testing. It consists of avoiding chip and software countermeasures with the use of focused ion beams, reverse engineering and circuit modification. Some of the most popular manufacturer countermeasures are the following:
    - Bus Scrambling
    - Memory Scrambling
    - Memory encryption
    - Metal Layers
    - Temperature sensors
    - Light sensors
    - Frequency sensors

[8]http://gauss.ececs.uc.edu/Courses/c653/lectures/SideC/intro.pdf
[9]http://www.engr.uconn.edu/~tehrani/teaching/tcs/sca_pa_shi.pdf
[10]https://www.riscure.com/archive/DPA_attack_on_RSA_in_CRT_mode.pdf
[11]https://www.riscure.com/documents/defeating_rsa_multiply-always_and_message_blinding_countermeasures.pdf?1378980229

3.7.1. Special SPA and DPA attacks

Simple Power Analysis (SPA) and Differential Power Analysis (DPA) were introduced by Kocher[9.]

SPA involves analyzing time-resolved electric current measurements directly, while DPA is based on statistical correlations between key bits and time-resolved current, and requires multiple runs to separate correlations from the background noise.

Large computational process such as DES rounds, RSA operations, etc, may be identified, since the operations performed by the microprocessor vary significantly during different parts of these operations. At higher magnification, individual instructions can be differentiated. SPA analyusis can, for example, be used to break RSA implementations by revealing differences betwhen multiplication and squaring operations which can be seen trough an Spectrum Analyzer.

4. Attacking the File Structure on Smart Cards.

Smart Cards manufacturers are known for adding backdoors and hidden information inside the devices. Manufacturers usually do not provide the complete card command reference manual and structure specification, being able to hide to the resellers and end users some features. Even, high end Smart Cards that are certified trough CC (Common Criteria), can contain hidden files or even administrative commands.

First of all, to discover all the commands available in a Smart Card, the Master File (MF) must be selected, which contains the actual application.

Later the application selection can be done through the "SELECT" command as stated in ISO/IEC 7816 Part 4.

Once selected the application, it is feasible to send trough the Smart Card reader a "SELECT" command for all possible Elementary Files (EF) that might exist. The "SELECT" command tries to retrieve the information held in the file, but if the actual user does not have permissions to access that file the card might response with a Status Word (SW) pointing as an access denied. With these error codes, access granted or denied we can discern which commands are valid or not.

The response codes are also part of the software uploaded in the Smart Card. If the card developer was savvy enough to forge special untrue error codes, this attack would not be successful at all.

The Standard ISO/IEC 7816 part 4 specifies that EF's can be retrieved with the command "00 A4 02 0C 02 ** **" where the four stars represent all possible values in hexadecimal from 0 to F. This means that there are 65536 possibilities to find EF's. This implies sending that amount of APDU's for each possible EF present in the card.

The overwhelming amount of APDU's required to automate this task. The author of this paper has elaborated a tool in Perl using the PCSC module to iterate trough all possible APDU's and to automatically discern which ones returns an invalid Status Word (SW).

As a proof of concept of this attack, an excerpt of the results provided by the tool is shown below.

```
sd@maindeb:~/Presentacio$ perl
smartattack.pl 02
+------------------------------------------
+
|Note: '-' mark used for: Incorrect
|    parameter
|Note: '.' mark used for: EF not found
|
+------------------------------------------
+

==[Start: Mon Feb  4 16:59:03
2013]======================
Test in progress ...
-
...........................................................
...........................................................
...........................................................
............................................
```

```
Found ICAO compliance selectable EF
id: 01 02
69 82

Found ICAO compliance selectable EF
id: 01 0e
69 82

Found ICAO compliance selectable EF
id: 01 1c
69 82
```

Looking at the output generated by the custom tool, we can see in green color the Elementary File (EF) found valid, and in red color, the Status Word (SW) code returned.

In this case, all SW are “69 82”, which as stated by the ISO/IEC 7816 Part 4 means “*Command not allowed - Security status not satisfied*”. This probably indicates that these commands exists, and shall be contrasted with the ones provided by the manufacturer (or the Standard itself) in order to identify undocumented Elementary Files (EF).

Attack vector mitigations:

Reducing the impact for this vulnerability is not trivial. File access on Smart Cards is difficult to control.

The only way to develop a countermeasure for this attack is coding it in the Operating System, as it can see all the file structure of the card. The card applications can only see its own files, so the countermeasure cannot be fully effective. In addition, counting how many times files are accessed and limit it to a certain amount is not an easy task. Cryptographic capable cards, ex: PKI purposes are very resources intensive, CPU, Crypto Processor and read-write operations. If dealing with long key algorithms and multi digital signing operations might render the card unusable if a counter is implemented.

The most plausible solution recommended would be to implement the counter in the Operating System level. That functionality shall have several profiles, which the card developer can choose. These profiles shall implement a larger or sherter counter for accessing files depending on the card uses. For example, for cryptographic intensive cards the counter boundary could be set to 100 file access per user session. For access control purpose cards, the counter shall be set to 10 per user session.

Bear in mind that in order to avoid card issues or rendering the card in non operational status, the counter shall be reset to ‘0’ once the user session is terminated or the card is powered off.

This is not an ultimate mitigation vector, but a partial, as if the developer sets a file access counter too low and the card goes beyond that limit the card can enter in a locked mode, be terminated or request an administrative user PIN, causing troubles to the users.

5. Attacking Commands definition on Smart Cards.

This attack is somehow similar to the previous, but with a different concept. Instead of "bruteforcing" EF, we do "bruteforce" APDU commands. As stated before, the card manufacturers can embed hidden commands to the card. In order to discover them, an approach is to execute sequentially all possibilities and compare the Status Word (SW) returned by the card.

As stated in chapter (3*.2. Commands definition***.**) APDU are composed of at least the following objects:

CLA + INS + P1 + P2

The remaining parameters to complete the APDU are optional.

To discover all possible hardcoded APDU's in the card we shall iterate all possibilities. There are 4 bytes, with representation in Hexadecimal; this means that altogether makes 4294967296 possible APDU commands.

In order to automate this task, the author has developed another module in Perl using the PCSC module to iterate trough all possible APDU commands and automatically contrast which response codes or Status Word (SW) returns the card.

This attack vector is also limited to the same effectiveness as the previous attack vector on Elementary Files (EF), as the Status Word is hardcoded in the card software, which might be altered by the developers.

To prove the effectiveness of this attack, an excerpt of the output generated by the tool is shown beneath:

```
sd@maindeb:~/Presentacio$ perl
smartattack.pl 03

APDU's BRUTEFORCING MIGHT
DAMAGE OR BLOCK YOUR CARD,
WANT TO CONTINUE? (y/n) : y

0: RSA PKCS-15 PKI application
1: WAP-WIM
2: FINEID
3: WAP-WIM
4: Girocard (Geldkarte) in Germany
5: Eurocheque card with chip in
Germany
6: com.gemplus.javacard.util packages
7: Gemplus card manager"
Description="434D = CM (ascii).
Security domain for some GCX/GXP
cards"
8: org.javacardforum.javacard.biometry
9: Global Platform Security Domain
AID
10: Machine Readable Travel
Documents (MRTD). Issuer stored data
application
11: Machine Readable Travel
Documents (MRTD). Application for
hashes, digital signature, and certificates

Please select number for app AID to
attack: 10
```

```
Want also to scan P1 and P2 recursively
(much more aggressive, ~4.200.000.000
tries instead of ~65.000)? y/n: n

P1 and P2 scanning is hardcore and
SmartCards countermeasures and
buffers gets full, want to activate
countermeasures to avoid the card
getting dizzy(higly recommended)? y/n:
y
Bruteforcing APDU's...



+-------------------------------------------+
|Note: '-' mark used to: indicate CLA rot
|     found
|Note: '.' mark used to: indicate INS not
|     found
+-------------------------------------------+

Choosen AID: [Machine Readable
Travel Documents (MRTD). Issuer
stored data application]
==[Start: Mon Feb  4 17:02:41
2013]======================
===========================
Test in progress ...
.............................................................
......
........................................
Found command: 00 70 00 00
................
Found command: 00 82 00 00
.
Found command: 00 84 00 00
```

To conduct this specific test a Spanish ePassport card was used. Once selected the AID of the ePassport with the following command (00 A4 04 0C 07 A0 00 00 02 47 10 01) is conducted a "bruteforce" attack over all APDU permutations. As a full scan of the card is time consuming and might render the card unusable, the script was stopped after some time.

Note the results marked in red color. The commands found by the tool were three: "00 70 00 00", "00 82 00 00", and "00 84 00 00". The results were compared with the ICAO specification, which extends the commands and structure of the ISO/IEC 7816 to standardize all ePassports. The conclusion about the legitimation of these commands seems appropriated.

Attack vector mitigations:

Reducing the impact for this vulnerability is not really tough at all. As all smartcards have a writable memory zone, it would be possible to code an algorithm that counts how many times an APDU instruction is sent and returned by the card to the reader, increasing the counter on each requests-response challenge per user session. When the user session is terminated or the card removed from the reader, the counter may be reset to '0'. That limit must be hardcoded in the card application (it would be also feasible to implement it in the Operating System of the card itself), and when the amount is equal or greater to that number the card itself can request an administrative user authentication, or simply terminate the card. Given the previous mitigation vector described, the recommended value for the counter depends on the SmartCard uses and applications installed in it. For an access card type for authentication purposes, the security counter shall be greater rather a card for shopping purposes. A counter of at least 20 APDU per user session would be a suitable counter measure to mitigate the attack.

6. Attacking the TRNG (True Random Number Generator).

Smart Cards capable of cryptographic operations implements the "GET CHALLENGE" command as defined in the ISO/IEC 7816 part 4, which supposedly returns a True Random Number generated by the card.

The numbers generated by the card are supposed to be unpredictable, as are used for key derivation, seed for symmetric encryption algorithms, Secure Channel establishment between the reader and the card, and many other critical operations.

Some other low end cards also implements the "GET CHALLENGE" command, but its result is not a real pure True Random Number (TRNG[12]). Instead is a Pseudo Random Number Generator (PRNG[13]), which is not as secure as the TRNG. The shortcomings of PRNG are that the seeds or the sources of these seeds are deterministic; not real random generated. These initial values come from a small pool of data not generated by a dedicated hardware random number generator.

Although they are pretty secure, they are not enough for critical PKI operations.

The purpose of this attack is to collect a sufficient quantity of random data from the card, pack it in a binary file, and statistically check it's randomness through the use of discrete probability distribution. To conduct this attack the Shannon Entropy test measure is used.

Further research revealed that already a tool to check entropy trough the information density in binary files exists, so no new code was to be written for such a purpose. This tool is called "*ENT*[14] *A Pseudorandom Number Sequence Test Program*".

The author of this paper has developed another module in Perl using the PCSC library to send the "GET CHALLENGE" command over and over the target card up to 4.000.000 times, in order to collect sufficient amount of random data to be statistically processed. The retrieved data is temporarily saved to an array, later saved on a text file, and then converted to a binary

[12] More Information about TRNG: http://en.wikipedia.org/wiki/Fortuna_%28PRNG%29

[13] More information about PRNG: http://en.wikipedia.org/wiki/Pseudorandom_number_generator

[14] ENT tool can be found here: http://www.fourmilab.ch/random/

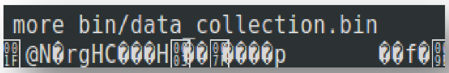
format file. Finally the Perl module automatically calls the "*ENT*" tool sending as a parameter the binary file generated.

To be able to instantiate the "GET CHALLENGE" command, first we must select the application AID as states the ISO/IEC 7816. To do so, we select it with the following command: "00 A4 04 0C 07 A0 00 00 02 47 10 01".

The card returns the Status Word (SW) "90 00", which means "Operation completed successfully".

Later the "GET CHALLENGE" command can be issued as many times as desired with the command "00 84 00 00 08".

The output generated by the Perl script is a binary file containing the random data as shown below:



```
more bin/data_collection.bin
@N�rgHC���H��������p        ��f��
```

Figure 3 – Content of the binary file containing random data collected from the card

As a proof of concept of this attack, an excerpt of the script that automatizes these tasks is shown below:

```
sd@maindeb:~/Presentacio$ perl
smartattack.pl 87
bin/binary_collection.bin

THIS IS A SIMPLE ENT (A
Pseudorandom Number Sequence Test
Program)
  Tool recommended configuration:

 +-------------------------------------------+
To pass the test minimum must achieve
7.976 bits per octet
+------------------------------------------+
Want to continue opening the tool?
(y/n): y

Entropy = 7.999974 bits per byte.

Optimum compression would reduce
the size
of this 19063096 byte file by 0 percent.

Chi square distribution for 19063096
samples is 678.94, and randomly
would exceed this value less than 0.01
percent of the times.

Arithmetic mean value of data bytes is
127.4913 (127.5 = random).
Monte Carlo value for Pi is
3.140673717 (error 0.03 percent).
Serial correlation coefficient is -
0.000016 (totally uncorrelated = 0.0).
```

The tool returned entropy of "*7.999974*" bits per byte, which is enough to be considered random as surpasses the minimum of 7.976 bits per octet.

Attack vector mitigations:

As suggested in the previous vulnerability (*4. Attacking Commands definition on Smart Cards.*) the approach to mitigate this vulnerability would be somewhat similar.

Cryptographic operations conducted in the crypto processors cannot be limited in any way. Doing so, would likely cause issues and are very specific for each algorithm and cryptographic key lengths employed. An asymmetric algorithm encryption process, for example RSA with a key length of 4096 bits would require a high crypto CPU use, but the same algorithm with key length of 1024 is way swifter.

Therefore, another way to mitigate this vulnerability shall be devised. The suggested approach in this research paper is to use again transaction counters. As explained before, counters cannot be implemented in cryptographic operations, but could in APDU commands.

Counting how many times a cryptographic function is requested per user session and setting a boundary trough a variable counter would be an effective way.

The operation counter shall be large enough to provide usability to the users and not rendering the card unusable.

Once the counter has reached the boundary, the card can be terminated, locked or to requests an administrative user authentication to unlock it.

7. Conclusions.

Smart Cards are supposed to be one of the most secure devices around, but the inclusion of possible backdoors, hidden commands, hidden files in the Operating System, and the uncertain seeds source of TRNG puts in check the overall security, including those already reviewed and certified with CC (Common Criteria) for government or military uses, which sometimes not enough source code analysis and low level testing is conducted.

A comprehensive testing shall be conducted on all certified cards, in order to search, among many other checks, the inclusion of hidden Elementary Files (EF), hidden operating system commands with tampered and non-compliant Status Words (SW) codes, and further cryptographic testing. Hardware backdoors also exists, which through the use of Logic Gates[15] an “evil” manufacturer can implement sequences to dump the entire card memory under specific circumstances.

The vulnerability mitigations stated in this document are an approach to increase the overall smart cards security against contrasted and real tested vulnerabilities.

A high level summary for each vulnerability is described as follows:

[15] More information about Logic Gates here: http://en.wikipedia.org/wiki/Logic_gate

<u>Attacking the File Structure on Smart Cards:</u>

- Developing a transaction counter in the card application is not feasible. Shall be done in the Operating System.

- Develop a file query counter to supervise the (ab)use of select file instructions.

- Precautions must be taken in case the card makes an extensive file write/access operations to avoid surpassing the counter limit, as could lock the card.

- Not recommended for High end Cryptographic cards (Ex. PKI uses)

- Counter must be reset to ‘0’ on power-off or user session expiry.

<u>Attacking Commands definition on Smart Cards.</u>

- Developing a transaction counter inside the card application or in the Operating System is adequate.

- Counter is suitable for almost all card types and uses.

<u>Attacking the TRNG (True Random Number Generator).</u>

- Limiting the Cryptographic operations processed by the Crypto processor is not a good way to mitigate the vulnerability. Large key length employed might cause issues with the counter.

- A transaction counter can be coded inside the Operating System of the card. This counter shall limit the recurrent and the consecutive calls to cryptographic instructions.

- This partial vulnerability mitigation might cause card issues (block the card) when performing strong cryptographic arithmetic’s.

The following table shows for each attack vector covered in this paper, the Mitigation difficulty, which measures the difficulty of implementing new countermeasures to solve the vulnerability. The Exploitation like hood which means how easy or difficult is for the average user to exploit the described vulnerability. Finally, the last row called Mitigation conclusion takes in account all the other rows and the mitigation vectors suggested to estimate the solutions effectiveness.

Vulnerability covered	Mitigation difficulty	Exploitation like hood	Mitigation conclusion
Attacking the File Structure on Smart Cards	**High**	**High**	**Partial**
Attacking Commands definition on Smart Cards	**Low**	**High**	**Complete**
Attacking the TRNG (True Random Number Generator)	**Medium**	**Medium**	**Partial**

8. Further Research.

Further research shall be done to cover and fully mitigate the stated vulnerabilities. With global market inclusion of new programming techniques and new hardware to produce Smart Cards, better techniques would be feasible.

An approach to further research on solving the first vulnerability, (Attacking the File Structure on Smart Cards) consists on devising a better way for the card itself to detect whether the internal file queries are legitimate or are an attack. A smarter logic can be coded inside the card to detect and block this attacks, but the counterpart of this proposal is the big extra effort that the CPU is subject to, which might be incompatible for payment solutions.

The second vulnerability, (Attacking Commands definition on Smart Cards), with the proposed mitigation vector, shall be good enough to solve the issue. Also it is not hard to develop and incorporate it in the card code.

Finally, the last vulnerability, (Attacking the TRNG (True Random Number Generator) is the most controversial to solve, as there is no easy approach to detect crypto attacks. Counting the recurrent access to crypto commands and resources in a very short period of time and smartly denying or allowing access would be an effective way to mitigate this vulnerability.

In the end, with the inclusion of newer technologies and protocols such as NFC new attack vectors rises, which unfortunately are beyond the scope of this paper**.**

9. Compatible cards tested during this PoC

The following table shows some cards tested during the evaluation and vulnerability testing conducted to develop this paper.

CARD ATR	Manufacturer	ISO-7816 Compatible	Advanced Crypto.
3B 04 07 3C 85 92	Telefonica	✓	✗
3B 04 07 3C 85 9A	Telefonica	✓	✗
3B 04 00 00 00 00	DNIe	✓	✓
3B 26 00 11 04 5C 03 90 00	La Caixa	✓	✓
3B 26 00 11 06 23 03 90 00	SG	✓	✗
3B 3B 11 00 6A 38 20 00 00 27 A0 33 33 90 00	Carrefour SIM	✓	✓/✗
3B 80 80 01 01	ePassport	✓	✓

10. Acknowledgments

The author would like to thank the reviewers for their helpful comments and advice. Finally, thanks to my family and friends on both continents who helped me in ways unknown to them.

11. Acronyms and definitions.

Smart Card: Integrated circuit capable of storing data securely

ICAO: International Civil Aviation Organization

CC: Common Criteria

TRNG: True Random Number Generator

PRNG: Pseudo Random Number Generator

OS: Operating System

Random Seed: number used to initialize a PRNG

NFC: Near field communication

APDU:	Application Protocol Data Unit
PKI:	Public Key Infrastructure
ePassport:	Electronic Passport
Hexadecimal:	Mathematical numeral system representation
SW:	Status Word
PCSC:	Personal Computer/Smart Card
EF:	Elementary File
MF:	Master File
DF:	Dedicated File
ISO:	International Organization for Standardization
PIN:	Personal Identification Number

12. References

[1.] Wolfgang Rankl, Wolfgang Effing, (2004), Smart Card Handbook, 3rd Edition 2005-01-15, ISO/IEC INTERNATIONAL STANDARD 7816 Part. 4. Second edition.

[2.] P. Kocher, ^{a}TimingAttacks on Implementations of Diffie-Hell-man, RSA, DSS, and Other Systems, Proc. Advances in Cryptology (CRYPTO '96),pp. 104-113, 1996

[3.] ISO7816-2 Identification Cards Integrated Circuit(s) Cards with Contacts Part 2: Dimensions and Location of the Contacts. Int'l Organization for Standardization, 1999.

[4.] ISO7816-3 Identification Cards Iintegrated Circuit(s) Cards with Contacts Part 3: Electronic Signals and Transmission Protocols. Int'l Organization for Standardization, 1997.

[5.] ISO7816-4 Identification Cards Integrated Circuit(s) Cards with Contacts Part 4: Inter-Industry Commands for Interchange.1995

[6.] ISO7816-7 Identification Cards Integrated Circuit(s) Cards with Contacts Part 7: Inter-Industry Commands for Structured Card Query Language (SCQL). Int'l Organization for Standardization, 1999

[7.] ISO/IEC 14443 Identification Cards Contactless Integrated Circuit(s) Cards Proximity Cards Part 4: Transmission Protocol. Int'lOrganization for Standardization, 2001.

[8.] J. Daemen and V. Rijmen, Resistance against Implementation Attacks: A Comparative Study of the AES Proposals, Proc.Second AdvancedEncryptionStandardCandidateConf., Mar. 1999. Available at http://www.nist.gov/aes.

[9.] P. Kocher, J. Jaffe, and B. Jun, Introduction to Differential Power Analysis and Related Attacks, http://www.cryptography.com/dpa/technical, 1998

[10.] ISO/IEC. ISO information, from:

http://www.iso.org/iso/home/about.htm

[11.] Tolga.K. (2001). from: http://www.tldp.org/HOWTO/Smart-Card-HOWTO/index.html

[12.] C. E. SHANNON, (1948) from: http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf

[13.] Compatible cards lists: http://ludovic.rousseau.free.fr/softwares/pcsc-tools/smartcard_list.t

[14.] Attacking algorithms from: http://www.cse.msstate.edu/~ramkumar/tamper2.pdf