Joint Interpretation Library

Application of Attack Potential to Smartcards

Version 2.1
April 2006

This page is intentionally left blank

Table of contents

1 Introduction

1 This document interprets the current version of Common Criteria Methodology [CEM], part 2, annex B.8. This work has been based on smartcard CC evaluation experience and input from smartcard industry through International Security Certification Initiative (ISCI)

2 This chapter provides guidance metrics to calculate attack potential required by an attacker to effect an attack. The underlying objective is to aid in expressing the total effort required to mount a successful attack. This should be applied to operational behaviour of a smartcard and not to applications specific only to hardware or software.

2 Scope

3 This document introduces the notion of an attack path comprised of one to many attack steps. Analysis and tests need to be carried out for each attack step on an attack path for a vulnerability to be realised. Where cryptography is involved, the Certification Body should be consulted.

3 Identification of Factors

3.1 How to compute an attack

4 Attack path identification and exploitation analysis and tests are mapped to relevant factors: elapsed time, expertise, knowledge of the TOE, access to the TOE, equipment needed to carry out an attack. Even if the attack consists of several steps identification and exploitation need only be computed for the entire attack path.

5 The identification part of an attack corresponds to the effort required to create the attack, and to demonstrate that it can be successfully applied to the TOE (including setting up or building any necessary test equipment). The demonstration that the attack can be successfully applied needs to consider any difficulties in expanding a result shown in the laboratory to create a useful attack. For example, where an experiment reveals some bits or bytes of a confidential data item (such as a key or PIN), it is necessary to consider how the remainder of the data item would be obtained (in this example some bits might be measured directly by further experiments, while others might be found by a different technique such as exhaustive search). It may not be necessary to carry out all of the experiments to identify the full attack, provided it is clear that the attack actually proves that access has been gained to a TOE asset, and that the complete attack could realistically be carried out. One of the outputs from Identification is assumed to be a script that gives a step-by-step description of how to carry out the attack – this script is assumed to be used in the exploitation part.

6 Sometimes the identification phase will involve the development of a new type of attack (possibly involving the creation of new equipment) which can subsequently be applied to other TOEs. In such a case the question arises as to how to treat the elapsed time and other parameters when the attack is reapplied. The interpretation taken in this document is that the development time (and, if relevant, expertise) for identification will include the development time for the initial creation of the attack until a point

determined by the relevant Certification Body. Once a Certification Body has determined this point, then no points for the development of the attack (in terms of time or expertise) will be used in the attack potential calculation.

7 The exploitation part of an attack corresponds to achieving the attack on another instance of the TOE using the analysis and techniques defined in the identification part of an attack. It is assumed that a different attacker carries out the exploitation, but that the technique (and relevant background information) is available for the exploitation in the form of a script or set of instructions defined during the identification of the attack. The script is assumed to identify the necessary equipment and, for example, mathematical techniques used in the analysis. This means that the elapsed time, expertise and TOE knowledge ratings for exploitation will sometimes be lower for exploitation than for identification. For example, it is assumed that the script identifies such things as the timing required for a perturbation attack, and hence in the exploitation phase the attacker does not have to spend significant time to find the correct point at which to apply the perturbation. Furthermore this same information may also reduce the exploitation requirement to one of time measurement, whereas the identification phase may have required reverse engineering of hardware or software information from power data – hence the expertise requirement may be reduced. Similarly, knowledge about the application that was used to achieve the timing of an attack may also be included either directly in the script or indirectly (through data on the timing required).

8 In many cases, the evaluators will estimate the parameters for the exploitation phase, rather than carry out the full exploitation. The estimates and their rationale will be documented in the ETR.

9 When presenting the attack potential calculation in the ETR, the evaluators will make an argument for the appropriateness of the parameter values used, and will therefore give the developer a chance to challenge the calculation before certification. The final attack potential result will therefore be based on discussions between the developer, the ITSEF and the CB, with the CB making the final decision if agreement cannot be reached.

10 It is an assumption of this interpretation that the Certification Bodies will ensure that there is harmonisation between national schemes. This is required, for example, where new types of attack are applied and a decision has to be taken as to when the attack is considered ‘mature’, at which point it will no longer gain point for the time or expertise to develop the attack (as discussed above).

3.2 Elapsed Time

11 Additional granularity is introduced into CEM elapsed time. In particular, distinction is drawn between one week and several weeks. Time is divided into the following intervals:

	Identification	Exploitation
< one hour	0	0
< one day	1	3
< one week	2	4
< one month	3	6
> one month	5	8
Not practical	*	*

Table 1: Rating for Elapsed Time

12 The CEM defines the term *Not Practical* as “the attack path is not exploitable within a timescale that would be useful to an attacker”.

13 In practice an evaluator is unlikely to spend more than 3 months attacking the TOE. At the end of the evaluation the evaluator has to assess the time it would take to carry out the minimum attack path. This computes the estimated time to mount the attack, and not necessarily the time spent by the evaluator to conduct the attack.

14 Where the attack builds on the findings of a previous evaluation, Elapsed Time as well as Expertise have to be taken into account, e.g. a particular attack may have been developed on a smartcard product similar to the TOE. It is not possible to give general guidance.

15 The question of "Not Practical" may depend on the specific attack scenario as the following two examples show:

(a) Assume a smartcard used for an online system, where the card contains only individual keys and assume further that these keys are deactivated in the system within days after loss of a card was reported. In this case an attack is not even practical for an attacker if he can extract the keys in one week.

(b) Assume a smartcard, which contains system-wide keys, which might be used for fraud even if use of the individual card is blocked after loss. In this case an attack may be successful for the attacker even if it takes a year.

16 So if a general assumption on a time for "Not Practical" is needed, something about 3-5 years is a better worst-case oriented time frame. (This is the time after which a card generation is normally exchanged and system wide keys may be changed in a comparable time frame). However, the best rule seems to decide on the meaning of "Not practical" only in a specific attack scenario.

3.3 Expertise

17 For the purpose of smartcards two types of experts are defined:

- an expert with the ability to define new attacks for smartcards (hardware, software, cryptography) and the necessary tools, and

- an expert with a commensurate level of knowledge of the TOE to that of the developer (e.g. knowledge of product standards and specifications).

18 Expertise necessary to carry out an attack may cover several disciplines: chemical, ability to drive sophisticated tools, cryptographic.

	Definition according to CEM	**Detailed definition to be used in smartcard evaluations**
a) Experts	Familiar with implemented • Algorithms • Protocols • Hardware structures • Principles and concepts of security	Familiar with • Developers knowledge namely algorithms, protocols, hardware structures, principles and concepts of security and • Techniques and tools for the definition of new attacks
b) Proficient	Familiar with • security behaviour	Familiar with • security behaviour, classical attacks
c) Laymen	No particular expertise	No particular expertise

Table 2: Definition of Expertise

Extent of expertise (in order of spread of equipment or smartcard related knowledge)	
Equipment: The level of expertise depends on the degree to which tools require experience to drive them • Optical Microscope • Chemistry (etching, grinding), Microprober • Laser Cutter, Radiation • Plasma (etching, grinding), Focused Ion Beam (FIB) • Scanning Electron Microscope (SEM), • Atomic Force Microscope (AFM)	**Knowledge:** The level of expertise depends on knowledge of • Common Product information • Common Algorithms, Protocols • Common Cryptography • Differential Power Analysis (DPA), Differential Fault Analysis (DFA), Smartcard specific hardware structures, Principles and concepts of security • Developers knowledge

Table 3: Extent of expertise

19 It may occur that for sophisticated attacks, several types of expertise are required. In such cases, the higher of the different expertise factors is chosen.

20 A new level “Multiple Expert” was introduced to allow for a situation, where different fields of expertise are required at an Expert level for distinct steps of an attack. It should be noted that the expertise must concern fields that are strictly different like for example HW manipulation and cryptography.

	Identification	Exploitation
Layman	0	0
Proficient	2	2
Expert	5	4
Multiple Expert	7	6

Table 4: Rating for Expertise

3.4 Knowledge of TOE

21 The CEM states “to require sensitive information for exploitation would be unusual”, however it shall be clearly understood that any information required for identification shall not be considered as an additional factor for the exploitation.

22 Since all sensitive and critical design information must be well controlled and protected by the developer, it may not be obvious how it assists in determining a dedicated attack path. Therefore, it shall be clearly stated in the attack potential calculation why the required critical information cannot be substituted by a related combination of time and expertise, e.g a planning ingredient for a dedicated attack.

23 The following classification is to be used:

- Public: this is information in the public domain,
- Restricted: this corresponds to assets which are passed about during the various phases of smartcard development. Suitable examples might be the functional specification (ADV_FSP), guidance documentation (AGD) or administrative documents usually prepared for smartcard issuers/customers. (See [CC-IC])
- Sensitive: HLD and LLD information.
- Critical: Implementation representation (Design and Source Code).
- Very critical hardware design: The designs of modern ICs involves not only huge data bases but also sophisticated bespoke tools. Therefore the access to useful data requires an enormous and time consuming effort which would make detection likely even with the support from an insider. If an attack is based on such knowledge the new level of “Very critical design” is introduced. It has to be decided in a case by case decision, if the knowledge can not be gained in another way.

24 In this way knowledge shall distinguish between access to high level design, low-level design on the one hand and source code/ schematics of the product on the other by taking into account two types of information (HLD/LLD and Implementation Level). (See [CC-IC])

25 It may occur that for sophisticated attacks, several types of knowledge are required. In such cases, the higher of the different knowledge factors is chosen.

	Identification	Exploitation
Public	0	0
Restricted<	2	2
Sensitive	4	3
Critical	6	5
Very critical hardware design	9	na

Table 5: Rating for Knowledge of TOE

3.5 Access to TOE

26 Availability of samples (in terms of time and cost) needs to be taken into account as well as the number of samples needed to carry out an attack path (this shall replace the CEM factor “Access to TOE“).

27 The attack scenario might require access to more than one sample of the TOE because:

- the attack succeeds only with some probability,
- the attacker needs to collect information from several copies of the TOE. In this case, TOE access is taken into account using the following rating:

	Identification	Exploitation
< 10 samples	0	0
< 100 samples	2	4
> 100 samples	3	6
Not practical	*	*

Table 6: Rating for Access to TOE

28 Not practical is explained as following:

- For identification: not practical starts with 2000 samples or the largest integer less than or equal to n/(1+(log n)^2), n being the estimated number of products to be built.
- For exploitation: not practical starts with 500 samples or the largest integer less or equal to n/(1+(log n)^3), n being the estimated number of products to be built.

29 The Security Policy as expressed in the Security Target should also be taken into account.

3.6 Equipment

30 In order to clarify equipment category, price and availability has to be taken into account.

- None
- Standard
- Specialized (this type of equipment shall be considered as the type of expensive equipment which universities have in their possession.)
- Bespoke
  - Expensive [CEM]
  - Difficult to keep confidential [CEM] such as PC's linked across Internet.

31 In an ideal world definitions need to be given in order to know what are the rules and characteristics for attributing a category to an equipment or a set of equipment. In particular, the price, the age of the equipment, the availability (publicly available, sales controlled by manufacturer with potentially several levels of control, may be hired) shall be taken into account. The tables below have been put together by a group of industry experts and will need to be revised from time to time.

32 The range of equipment at the disposal of a potential attacker is constantly improving, typically:

- Computation power increase
- Cost of tools decrease
- Availability of tools can increase
- New tools can appear, due to new technology or to new forms of attacks

33 It may occur that for sophisticated attacks, several types of equipment are required. In such cases by default the higher of the different equipment factors is chosen.

3.7 Tools

34 The border between standard, specialized and bespoke can not be clearly defined here. The rating of the tools is just a typical example. It is a case by case decision depending on state of the art and costs involved. The following tables are just a general guideline.

Tool	Equipment
UV-light emitter	Standard
Climate chamber	Standard
Voltage supply	Standard
Oscilloscope analogue	Standard
Chip card reader	Standard
PC or work station	Standard
Signal analysis software	Standard

Tool	Equipment
Signal generation software	Standard
Visible light microscope and camera	Specialized
UV light microscope and camera	Specialized
Micro-probe Workstation	Specialized
Laser equipment	Specialized
Signal and function processor	Specialized
Oscilloscope digital	Specialized
Signal analyzer	Specialized
Tools for chemical etching (wet)	Specialized
Tools for chemical etching (plasma)	Specialized
Tools for grinding	Specialized

Table 7: Categorisation of Tools (1)

3.7.1 Design verification and failure analysis tools

35 Manufacturers know the purchasers of these tools and their location. The majority of the second hand tools market is also controlled by the manufacturers.

36 Efficient use of these tools requires a very long experience and can only be done by a small number of people. Nevertheless, one cannot exclude the fact that a certain type of equipment may be accessible through university laboratories or equivalent but expertise in using the equipment is quite difficult to obtain.

Tool	Equipment
Scanning electron microscope (SEM)	Bespoke
E-beam tester	Bespoke
Atomic Force Microscope (AFM)	Bespoke
Focused Ion Beam (FIB)	Bespoke
New Tech Design Verification and Failure Analysis Tools	Bespoke

Table 8: Categorisation of Tools (2)

37 Note, that using bespoke equipment should lead to a moderate potential as a minimum.

38 The level “Multiple Bespoke” is introduced to allow for a situation, where different types of bespoke equipment are required for distinct steps of an attack.

	Identification	Exploitation
None	0	0
Standard	1	2
Specialized (1)	3	4
Bespoke	5	6
Multiple Bespoke	7	8

Table 9: Rating for Equipment

(1) If clearly different testbenches consisting of specialised equipment are required for distinct steps of an attack this shall be rated as bespoke.

39 Equipment can always be rented but the same quotation applies.

3.8 Open Samples/Samples with known Secrets

3.8.1 Definition of "open samples / Samples with known Secrets"

40 The term "open samples" stands for samples were the evaluator can put SW on the HW platform on his own discretion. The intention is to use test SW without SW countermeasures but not deactivate any IC inherent countermeasures. The SW should serve to highlight IC properties described in the IC ETR-LITE considering the special use of the HW in the TOE but not repeat the IC evaluation. If the IC allows different configurations, the configuration implemented in the TOE shall be used. With these samples, it is possible to characterise the HW without SW.

41 "Samples with known secrets" refers to a TOE for which the evaluator knows or can define one or more pieces of secrets data, such as PIN or key.

3.8.2 Use of "open samples / Samples with known Secrets"

42 For a composite evaluation, the TOE is the combination of HW and SW and the attacks during the evaluation have to be directed against this combination. For the definition of the attacks, the evaluator has to have full knowledge of the TOE. For the HW part in a composite evaluation this knowledge is provided by the evaluation results as described in the CC supporting document [COMPO].

43 The documents passed on from the HW evaluation to the composite evaluator describe the protection against threats and states requirements on the environment (especially the SW) necessary to obtain this protection. In addition, the documents will be guidance on how the HW has to be used to achieve the security objectives.

44 For the vulnerability analysis and definition of attacks he wants to perform, the evaluator of the composite TOE can build on this information.

45 In special cases the vulnerability analysis and definition of attacks might be difficult, need considerable time and require extensive pre-testing if only this information is available. For example, samples with known secrets will allow faster characterization and allow a clear demonstration of successful attacks.

46 The platform may be used in a way that was not foreseen by the HW developer and evaluator or the SW provider may not have followed the recommendations provided with the HW.

47 The composite evaluator has to consider parts of the HW functionality that may not have been covered by the security target of the HW and therefore the HW evaluation.

48 Different possibilities exist to shorten the evaluation time in such cases:

- The composite evaluator can consult the evaluator of the HW and draw on his experience gained during the evaluation
- Separation of vulnerabilities of SW and HW with the use of "open samples"

3.8.3 Implications on the composite evaluation

49 With the use of "open samples", it is possible to factorise attack paths and by that reduce the complexity of an attack. That saves time in the evaluation because it makes it possible to obtain the targeted result much faster.

50 A good example for this is the retrieving of secret information (e.g. keys) by light attacks. In a well-designed product, the HW as well as the SW will have protective mechanisms to avert this attack. In combination, they will make attacks quite difficult. The evaluator will have to try a very high number of combinations and variations of parameters like beam diameter, light frequency, light strength, location for applying the light, position in time for the light flash. This gets especially difficult if the SW contains means to render the TOE inoperable if an attack is detected. An attack could not only prove very time consuming but also require a great number of samples.

51 With "open samples", the situation is quite different. The evaluator can use his own optimised test program and scan the IC for "weak spots" much faster and without risking the destruction of the device (the fact that such "weak spots" exist might even have been stated in the HW evaluation documentation). With the knowledge gained in these tests the attacker can then launch much more directed attacks on the TOE.

52 This example also shows the danger of this approach. Without open samples, the attack on the TOE (combination of HW + SW) might be not realistic and unfeasible. Therefore, this would lead to unjustified rating and in the extreme to a fail of the product.

3.8.4 Implications on the composite rating

53 For the rating two possibilities have to be considered:

- Freely programmable samples of the HW or similar variants are freely available. In that case, the samples are not to be considered as "open samples". They have to be considered just a tool (like e.g. a microscope) for the evaluator. The results can be used without any special treatment in the rating.
- The access to the samples is restricted and controlled and has been evaluated during the IC evaluation. In that case, the rating has to include an additional factor for the use of "open samples" as described in the table below.

3.8.5 Calculating the attack potential

54 Using open samples the evaluator will perform two calculations:

- Estimating the value for each factor for an attacker without access to open samples.
- Giving the values for each factor corresponding to what he as done:
  - Time spent, destroyed samples, Expertise, Knowledge of the TOE, equipment
  - Adding the points corresponding to the open samples used

55 The final value will be the minimum of the two calculations. It is expected that the two values are quite close. If this is not the case further analysis is required to decide on the rating.

56 The points corresponding to the availability of open samples are defined by taking into account the protection and the control of these open samples during the entire live cycle.

57 For ICs, the protection level will be analysed during the IC evaluation and stated in the ETR LITE.

58 For “samples with known secret”, defining the protection level is part of the evaluation of the full product.

59 Because of the similarity in the in the threat to the TOE, the rating should be defined according to the values of the Knowledge of the TOE factor: PUBLIC, RESTRICTED, SENSITIVE and CRITICAL:

- PUBLIC: No protection of the samples, delivered without control (no NDA, no checking of the customer); or the IC is used in non-secure applications (e.g. applications without guarantee of implementing the security recommendations or versions which can be freely programmed with native code).
- RESTRICTED: Typically protected as the specifications of the card, as the data sheet of an IC, or delivered without extra control of the people having access to this kind of information.
- SENSITIVE: Protected as the HLD/LLD design levels are.
- CRITICAL: Protected as the implementation level (source code, VHDL, layout). This requires to have very few open samples produced, to have very strong control of their delivery and to have the assurance that the receiving organisation is able to setup a control at the same level.

60 The composite evaluation has also to define if the use of “open samples” **and** “samples with known secret” adds the same time and add points for each of them. The analysis will be done during the ALC_DVS.2 task, checking if a **single collusion can be enough or if two different collusions** are necessary.

61 **The IC evaluation will give a rating for the “open samples” in the ETR-LITE. Any indication for a different rating has to be considered in the composite evaluation.**

Access to open samples	Identification	Exploitation
Public	0	
Restricted<	2	
Sensitive	4	
Critical	6	

3.9 Final Table

Factors	Identification	Exploitation
Elapsed time		
< one hour	0	0
< one day	1	3
< one week	2	4
< one month	3	6
> one month	5	8
Not practical	*	*
Expertise		
Layman	0	0
Proficient	2	2
Expert	5	4
Multiple Expert	7	6
Knowledge of the TOE		
Public	0	0
Restricted	2	2
Sensitive	4	3
Critical	6	5
Very critical hardware design	9	na
Access to TOE		
< 10 samples	0	0
< 100 samples	2	4
> 100 samples	3	6
Not practical	*	*
Equipment		
None	0	0
Standard	1	2
Specialized (1)	3	4
Bespoke	5	6
Multiple Bespoke	7	8
Open samples (rated according to access to open samples)		
Public	0	na

Factors	Identification	Exploitation
Restricted<	2	na
Sensitive	4	na
Critical	6	na

Table 10: Final table for the rating factors

(1) If clearly different testbenches consisting of specialised equipment are required for distinct steps of an attack this shall be rated as bespoke.

* Indicates that the attack path is not exploitable within a timescale that would be useful to an attacker. Any value of * indicates a High rating.

62 The following table replaces table B.4 of CEM, para 1873 for smartcards.

Range of values	Resistance to attacker with attack potential of:	SOF rating
0-15	No rating	No rating
16-24	Low	Basic
25-30	Moderate	Medium
31 and above	High	High

Table 11: Rating of vulnerabilites

4 Examples of attack methods

63 The following examples have been compiled by a group of security experts representing the different actor groups involved in the development, production, security evaluation and distribution of a smartcard product (Hardware vendors, Card vendors, OS provider, Evaluation labs, Certification bodies, Service providers**).**

64 The collection represents the current state of the art at that time (Q4/05). As state of the art is not static this document is under review of the same expert group and will be updated if necessary.

65 For the evaluation of a TOE at least these examples have to be considered. This does not mean that in any case all attacks have to be carried out. For each TOE the evaluation lab conducting the evaluation has to select the appropriate attacks from this catalogue in agreement with the certification body. This selection will be dependent on the type of the TOE and additional tests may also be required.

66 In this document only a general outline of the attacks is given. For more detailed descriptions and examples, please refer to the certification bodies. They can also provide examples as reference for rating.

4.1 Physical Attacks

67 Microelectronic tools enable to either access or modify an IC by removing or adding material (etching, FIB, etc). Depending on the tool and on its use the interesting effect for the attacker is to extract internal signals or manipulate connections inside the IC by adding or to cutting wires inside the silicon.

68 Memories could also be physically accessed for, depending on the memory technology, reading or setting bit values.

69 The attack is directed against the IC and often independent of the embedded software (i.e. it could be applied to any embedded software and is independent of software counter measures).

70 The main impacts are:

- Access to secret data such as cryptographic keys (by extracting internal signals)
- Disconnecting IC security features to make another attack easier (DPA, perturbation)
- Forcing internal signals
- Even unknown signals could be used to perform some attacks

71 The potential use of these techniques is manifold and has to be carefully considered in the context of each evaluation.

4.2 Overcoming sensors and filters

72 This attack covers ways of deactivating or avoiding the different types of sensor that an IC may use to monitor the environmental conditions and to protect itself from

conditions that would threaten correct operation of the TOE. Hardware or software may use the outputs from sensors to take action to protect the TOE.

73 Sensors and filters may be overcome by:

- Disconnection
- Changing the behaviour of the sensor
- Finding gaps in the coverage of the monitored condition (e.g. voltage), or of the timing of monitoring.

74 Sensors may also be misused, in order to exploit activation of a sensor as a step in an attack. This misuse of sensors is a separate attack.

75 The different types of sensors and filters include:

- Voltage (e.g. high voltage or voltage spike)
- Frequency (e.g. high frequency or frequency spike)
- Temperature
- Light (or other radiation)

76 The main impacts are:

77 The correct operation of a chip can no longer be guaranteed outside the safe operating conditions. The impact of operating under these conditions may be of many sorts. For example:

- Contents of memory or registers may be corrupted
- Program flow may be changed
- Failures in operations may occur (e.g. CPU, coprocessors, RNG)
- Change of operating mode and/or parameters (e.g. from user to supervisor mode)
- Change in other operating characteristics (e.g. changed leakage behaviour; enable other attacks like RAM freezing, electron beam scanning).

78 If a chip returns incorrect cryptographic results then this may allow a DFA attack, see section 4.4. Other consequences are described under general perturbation effects in section 4.3

4.3 Perturbation Attacks

79 Perturbation attacks change the normal behaviour of an IC in order to create an exploitable error in the operation of a TOE. The behaviour is typically changed either by applying an external source of energy during the operation of the IC, or by operating the IC outside its intended operating environment (usually characterised in terms of temperature, Vcc and the externally supplied clock frequency).

80 The attack will typically aim to make cryptographic operations weaker by creating faults that can be used to recover keys or plaintext, or to avoid or change the results of checks such as authentication or lifecycle state checks or else change the program flow.

81 Chapter 4.3 concerns itself more with the methods to induce meaningful faults whereas Chapter 4.4 describes how these induced faults may be used to extract keys from cryptographic operations.

82 Perturbations may be applied to either a hardware TOE (an IC) or a software/composite TOE (an OS or application running on an IC).

83 The main impacts are:

84 For attackers, the typical external effects on an IC running a software application are as follows:

- Modifying a value read from memory during the read operation: The value held in memory is not modified, but the value that arrives at the destination (e.g. CPU or coprocessor) is modified. This may concern data or address information.
- Changing the characteristics of random numbers generated (e.g. forcing RNG output to be all 1's) – see Attacks on RNG 4.9 for more discussion of attacks on random number generators.
- Modifying the program flow: the program flow is modified and various effects can be observed:
    - Skipping an instruction
    - Inverting a test
    - Generating a jump
    - Generating calculation errors

85 It is noted that it is relatively easy to cause communication errors, in which the final data returned by the IC is modified. However, these types of errors are not generally useful to an attacker, since they indicate only the same type of errors as may naturally occur in a communication medium: They have not affected the behaviour of the IC while it was carrying out a security-sensitive operation (e.g. a cryptographic calculation or access control decision).

86 The range of possible perturbation techniques is large, and typically subject to a variety of parameters for each technique. This large range and the further complications involved in combining perturbations means that perturbation usually proceeds by investigating what types of perturbation cause any observable effect, and then refining this technique both in terms of the parameters of the perturbation (e.g. small changes in power, location or timing) and in terms of what parts of software are attacked. For example, if perturbations can be found to change the value of single bits in a register, then this may be particularly useful if software in a TOE uses single-bit flags for security decisions. The application context (i.e. how the TOE is used in its intended operating environment) may determine whether the perturbation effect needs to be precise and certain, or whether a less certain modification (e.g. one modification in 10 or 100 attempts) can still be used to attack the TOE.

4.4 Retrieving keys with DFA

87 DFA is the abbreviation of Differential Fault Analysis. With DFA an attacker tries to obtain a secret by comparing a calculation without an error and calculations that do have an error. DFA can be done with non-invasive and invasive techniques.

88 This class of attacks can be divided in the following stages:

- Search for a suitable fault injection method
- Mounting the attack (performing the cryptographic operation once with correct and once with faulty parameters)
- Retrieving the results and composing a suitable set of data and calculating the keys from that data

89 By applying special physical conditions during the cryptographic operation, it is possible to induce single faults (1 bit, 1 byte) in the computation result.

90 This attack can be carried out in a non-invasive or an invasive manner. The non-invasive method (power glitching) avoids physical damages. The invasive method requires the attacker to physically prepare the TOE to facilitate the application of light on parts of the TOE.

91 The main impacts are:

92 DFA can break cryptographic key systems, allowing to retrieve DES, 3DES and RSA keys for example, by running the device under unusual physical circumstances. The attacker needs to inject an error at the right time and location to exploit erroneous cryptographic outputs.

93 As keys and code are usually present in EEPROM it might be difficult to randomly alter bits without crashing the entire system instead of obtaining the desired faulty results, although code alteration can give results as well. Other techniques may be useful to determine best location and time to inject an error; such as analyzing the power consumption to determine when the cryptographic computation occurs.

4.5 SPA/DPA – Non-invasive retrieving of secret data

94 SPA and DPA stand for ‘Simple’ and ‘Differential Power Analysis’, respectively, and aim at exploiting the information leaked through characteristic variations in the power consumption of electronic components – yet without damaging the TOE in any way what-so-ever. Although various levels of sophistication exist, the power consumption of a device can in essence be simply measured using a digital sampling oscilloscope and a resistor placed in series with the device. The outcome of the attack may be as simple as a characteristic trigger point for launching other attacks (such as DFA), or as much as the secret key used in a cryptographic operation itself. Depending on the goal of the attack it may involve a wide range of methods from direct interpretation of the retrieved signal to a complex analysis of the signal with statistical methods.

95 The main impacts are:

96 It lies in the very nature of SPA and DPA attacks that they may in principle be applied to any cryptographic algorithm – either stand alone, or as part of a composite attack. Additionally, SPA may serve as a stepping stone for launching further attacks. For instance, SPA may be employed to detect a critical write operation to the EEPROM that needs to be intercepted. An SPA analysis may also be performed as part of a timing attack, or for deducing which branch of a conditional jump has been taken by the program flow. Finally, an SPA attack could be used to determine the proper trigger point for a subsequent glitch or light attack, or as an aide for localising a suitable time window for a physical probing attack.

97 A DPA attack does not need to be entirely successful for it to become dangerous. Given a suitable key search strategy that takes into account imperfect DPA results, it may be enough to retrieve only part of the secret key by DPA, and obtain the rest by brute-force methods.

4.6 Higher Order DPA

98 Implementations that include countermeasures like boolean masking that resist first order DPA may be vulnerable to higher order DPA. This requires that the attacker is able to correlate more than once per TOE computation using hypotheses on intermediate states that depend on secret key parts.

99 The combined statistical analysis may be based on aligned measurements of the same side channel at different times or on aligned simultaneous measurements of different channels like power consumption and electromagnetic radiation of the device during the computation.

100 The attack requires at least the same effort as for a standard DPA attack with respect to expertise, knowledge of the TOE and equipment. Depending on the countermeasures and the implementation the effort increases at least for the identification of the attack.

101 The main impacts are:

102 Although higher order DPA is also a generic approach the analysis must be adapted for every TOE since the information that must be combined for the statistical analysis depends on the implementation including the combination of countermeasures. Nevertheless higher order DPA can be applied to different cryptographic systems such as all kind of secret key (symmetric) algorithms that make use of similar Boolean or arithmetic masking countermeasures. (Of course, algorithms that are vulnerable to first order DPA are vulnerable to higher order DPA too.)

103 The extension of higher order DPA to public key (asymmetric) algorithms seems to be very difficult because of the widely applied blinding measures that make use of algebraic transformations during the calculation that are completely different from masking. Therefore higher order DPA is more adapted and efficient when used to retrieve secret information from symmetric than asymmetric algorithms.

104 With higher order DPA, it may be possible to analyse implementations with countermeasures against standard DPA.

105 However, it seems that a non-negligible part of higher order DPA success is in parallel

- with the observer's experience,
- his ability to recognize and to interpret the different clues to follow,
- his ability to develop the corresponding tools that will be used to track this information,
- his knowledge and skills in cryptography and the analysis of the hardware design as well as
- with the environment conditions of the experiment itself,

106 in order to gain access to the sought-after information..

4.7 EMA Attacks

107 When an IC is operating, each individual element will emit electromagnetic radiation in the same way as any other conductor with a current flowing through it. Due to the change of the data processed, small changes in the current flow will be the result. These current flow changes lead to an electromagnetic emission depending on the processed data.

108 Electromagnetic Analysis (EMA) attacks measure these electromagnetic emissions from an IC during its operation and inferences to the data processed. It uses similar analysis techniques to those used in power analysis, hence it is sometimes referred to as SEMA (Simple Electromagnetic Analysis, analogous to Simple Power Analysis (SPA)) or DEMA (Differential Electromagnetic Analysis, analogous to Differential Power Analysis (DPA)).

109 The attack may use emissions from the whole IC (chip-EMA), or may focus on the emissions from particular areas of the die, where critical components are located (local-EMA).

110 Experimental evidence show that electromagnetic obtained data (particularly from localised areas of a die) can be different from power trace data, and ICs that are protected against power analysis may therefore be vulnerable to EMA.

111 The attack will typically aim to recover keys or plaintext, but may also be applied to recover other secret data such as PINs, or random numbers generated for use as secrets.

112 The main impacts are:

113 An EMA attack may be used in various ways – the following are examples:

- Used for determination of secret data correlated with emissions, such as keys or PINs, in a similar way to that of SPA or DPA (note that the EMA attack may be more efficient, requiring fewer examples, than a power-based attack)
- Used for identification of power analysis countermeasure activity, enabling them to be removed from power traces hence enabling a power analysis attack to succeed
- Used for identification of distinct localised activity that can be used in breaking security functions – for example, it might be possible to detect different register configurations for squares and multiplies in a coprocessor that supports RSA
- Used for identification of activity that may assist in synchronisation of other attacks – for example, it may be possible to detect actions within a cryptographic algorithm or PIN check that enable the precise synchronisation of a perturbation (see chapter 4.3 Perturbation Attacks).

114 As with power analysis, EMA attacks may be carried out for a hardware TOE (an IC), or a software/composite TOE (an OS or application running on an IC). In the same way as for power analysis, the vulnerability of software to EMA attack should not be predicted from the EMA characteristics of the hardware alone. The way in which software uses the IC functions may make a critical difference to its vulnerability to this type of attack.

4.8 Exploitation of Test features

115 The attack path aims to enter the IC test mode to provide a basis for further attacks.

116 These further attacks might lead to disclosure or corruption of memory content but as this depends on the possibilities of the test mode this is not considered here.

117 The main impacts are:

118 As result of a successful attack, the attacker is able to read out the content of the non-volatile memory using test functions. The implementation of the test functions may have an impact on the usability of the retrieved user data.

119 Another result is the re-configuration of life cycle data or error counters using a test function. Thereby an attacker is able to continue his analysis on the same device.

4.9 Attacks on RNG

120 Attacks on RNGs aim in general to get the ability to predict the output of the RNG (e.g. of reducing the output entropy) which can comprise:

- past values of the RNG output (with respect to the given and possibly known current values),
- future values of the RNG output (with respect to the possibly known past and current values),
- forcing the output to a specific behaviour, which leads to:
  - known values (therefore also allowing for the prediction of the output),
  - unknown, but fixed values (reducing the entropy to 0 at the limit),
  - repetition of unknown values either for different runs of one RNG or for runs of two or more RNGs (cloning) .

121 A RNG considered here can be one of the following types[1]:

- true RNGs (TRNG), the output of which is generated by any kind of sampling inherently random physical processes,
- pseudo RNG (PRNG) which output is generated by any kind of algorithmic processing (the algorithm is in general state based, with the initial state (seed) may generated by a TRNG),
- hybrid RNG (HRNG), which consists of a TRNG and a PRNG with a variety of state update schemes,

122 The applicable attack methods vary according to the Type of RNG:

123 A true RNG may be attacked by[2]:

- permanent or transient influence of the operating conditions (e.g. voltage, frequency, temperature, light)
- non invasive exploitation of signal leakage (e.g. signal on external electrical interfaces)

[1] In the context of smart cards the RNG based on some measurements of environment are not considered to be relevant.

[2] It is here assumed that the direct attack on a true RNG (i.e. guessing the value) is not feasible for any attacker.

- physical manipulation of the circuitry (stop the operation, force the line level, modify and/or clone the behaviour, disconnect entropy source)
- wire taping internal signals (compromise internal states)

124 A pseudo RNG may be attacked by:

- direct (cryptographic) attack on the deterministic state transition and output function (e.g. based on known previous outputs of the RNG)
- indirect attack on the state transition computation process by employing some side channel information (i.e. leakage on external electrical interfaces)
- attack on the execution path of the processing (modification of the results)
- attack on the seed (prevent reseeding, force the seed to fixed known or unknown (but reproducible) value, compromise the seed value)
- overcome the limit of RNG output volume (e.g. forcing the RNG to repeat values or to produce enough output to enable the attacker to solve equations and based on the solution to predict the output)

125 The attacks on hybrid RNG will be in general a combination of attacks on TRNGs and PRNGs.

126 All RNG designs can be expected to demand also for test procedures to counter attacks like those listed above. The analysis above does not take attacks on test procedures into account, as such attacks will by covered sufficiently by the more general attack scenario on software. Observe that test procedures may be an object on attack like SPA/DFA to reveal the RNG output values.

127 The main impacts are:

128 A successful attack on the RNG will result in breaching the security mechanisms of the chip, which rely on the randomness of the RNG. The mechanisms may be DPA/SPA countermeasures, sensor testing, integrity checking of active shield, bus and/or memory encryption and scrambling. The application software is affected by such attacks indirectly, e.g. sensors and related tests being disabled by an attacker, will generate further attack possibilities.

129 The software developer can rely on the capabilities of the hardware platform for testing the RNG and use these or implement and perform additional tests by himself based on such capabilities. The software developer may implement also tests for repetition of RNG output, but the coverage and feasibility of such tests may depend on the implementation and seems to be a problem. The cloning attack for RNG output on different instances of a RNG cannot be countered by tests, so other mechanisms must be designed as appropriate.

130 In case of TRNGs, sufficient tests should be performed (either by the chip platform itself or by the software developer). [AIS31] is an example of a methodology for assessing the effectiveness of the testing mechanisms. In case of PRNG a special effort on protecting the seed and the algorithm in terms of integrity and confidentiality is required. This effort pertains to the general software and data protection aspects and will be not discussed further in this chapter.

4.10 Ill-formed Java Card applications

131 This logical attack consists in executing **ill-formed applications**, i.e. malicious applications that are made of illegal sequences of byte-code instructions or that do not have valid byte-code parameters.

132 This example is only applicable to Java Cards (although there may be equivalent attacks for other operating systems). If not combined with any other attack such as authentication bypass, this attack has to be applied to Java Cards with known loading keys (these could be considered as open mode samples). In addition, if the card includes an embedded byte-code verifier, this verifier must be disabled. No other specific configuration is required.

133 Ill-formed applications execute a sequence of byte-code that violates the Java rules. Ill-formed applications are usually created from standard applications, in which the byte-code is manually modified. It means that such ill-formed applications cannot be the output of a normal CAP file generator. As a consequence, most Java Card platforms don't enforce the rules during the execution of applications.

134 The main impacts are:

135 In the most favourable cases, the attacker can retrieve information (e.g. a dump of memory), execute functions that usually require specific privileges or even switch to a context giving the full control over the card (JCRE context).

4.11 Software Attacks

136 Most of the examples of attacks in this document require hardware attack steps for all or part of the attack. However, it is clear that there are many relevant attacks that can be made on software alone. This section considers some of these attacks. In many cases software attacks start with source code analysis.

137 In general, it is important to note that most software attacks arise from errors (bugs) in the TOE, either in design or implementation. In these cases, the error will generally result in a failure to meet the requirements of one (or more) of the ADV families (e.g. ADV_IMP.1.2E: The evaluator shall determine that the least abstract TSF representation provided is an accurate and complete instantiation of the TOE security functional requirements). Hence an error of this sort will cause the TOE to fail evaluation (or, more usually, will require a modification to the TOE to correct the error).

138 In some other cases, a design's specification may be insufficient to meet the TOE security objectives: for example, a protocol specification might itself contain critical vulnerabilities. This would also cause a TOE to fail the evaluation.

139 This section therefore lists a number of attack steps that may be used to discover software errors, but no attack potential examples are given, since if any error is discovered then it must be corrected if the TOE is to pass evaluation.

140 In the text below we consider first an information gathering attack step, which may be relevant to a number of different types of attack. We introduce five specific attack techniques that may exploit software vulnerabilities:

- Editing commands
- Direct protocol attacks
- Man-in-the-middle attacks
- Replay attacks
- Buffer overflow

141 The attacks are of a logical nature, the test environment consists of a smart card reader connected to a PC. The PC runs communication software, a protocol analyser and some development tools to modify communication. This tool set is considered to be standard equipment. Tools are available as freeware on the Internet, and they can be modified quite easily to fit the attackers' needs.

142 To perform such attacks, it is necessary to have:

- a means to listen to message sequences (reader, traffic analyser)
- a means to create messages (information on external API, pattern generator)
- a means to interrupt messages without detection (protocol dependent)

143 Setting up a test environment and identifying an attack is quite simple, as the tools are standard and the commands are often ISO standard, and therefore public knowledge. If the command set is proprietary, the expertise needed is slightly higher because the communication must be interpreted. However, in most cases this would be expected to be relatively straightforward, and this type of 'security by obscurity' would not be considered a valid defence against attack.

4.12 Information gathering

4.12.1 Introduction

144 By their nature, communication protocols are susceptible to information leakage. This unwanted effect is a consequence of the fact that they are designed to pass information. This type of attack tries to use the protocols in ways that were not intended by the protocol developer, by first gathering information and then changing that communication to obtain secret data or other resources.

145 The attack step is usually a non-invasive technique, with the aim of getting information on the communication commands that the smartcard supports or using information from message sequences to enable other attacks. It is noted that the information is assumed to be information not contained in design documents (e.g. undocumented responses to commands). This information may then enable the attacker to modify the interaction or to disclose information (e.g. user data or keys) using weaknesses in the software implementation. This attack step is normally not a full attack path leading to the retrieval of secret data, although it might do in specific cases.

146 This attack step results in gathering information on the operation of the TOE, with possible disclosure of secret data (exposure of secret data in this way would generally be considered a sufficient vulnerability to cause the TOE to fail evaluation[3]). The information gathered is analysed to see whether it can be used to mount an attack to

[3] Depending on the scope of the evaluation and the environment, there may be some situations where such information exposure is accepted, e.g. in a protocol for use only in secure personalisation environments.

retrieve secret data from the TOE with one of the other mechanisms described in this document. The attacker knows the attack has succeeded by analysing the answers the smartcard gives during the communication.

4.12.2 Attack Step Descriptions

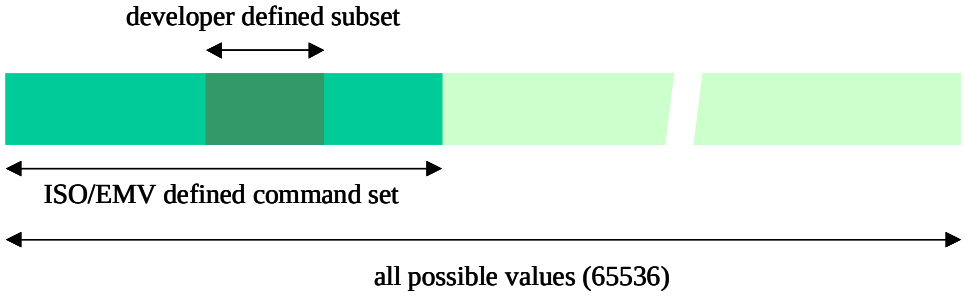
Observing Message Sequences

147 Observing message sequences may result in:

- obtaining information on an unknown protocol (e.g. where the interface specification is not public) to prepare an attack
- obtaining information on unknown internal product structures (typically data structures in software) to prepare an attack
- disclosing information, keys, or security attributes during import or export operations
- tracing product activity or user behaviour (e.g. to enable a replay attack).

Command searches

148 The total amount of values that a smartcard can communicate using a typical protocol such as ISO 7816 T=1 is 2^{16}, or 65536 different commands. Of this set, ISO defined a subset as being valid commands. And of this ISO set, a developer defines a subset and documents these commands as being valid commands for this card.



149 A T=1 test plan should contain the following tests:

- A ‘brute force’ approach in which all values outside the ISO defined set are tried and it is checked whether the card responds (inopportune behaviour).
- A ‘brute force’ approach in which all values of the ISO defined set, but outside the developer defined set are tried for a response (undocumented command search).
- Trying all developer documented commands and checking the answers.
- Influencing the communication by sending commands in different sequences.
- Interrupting message from system or from product

150 Attacks that make use of undocumented commands and editing commands are closely related, but distinctive attacks. Finding undocumented or undefined commands is a straightforward brute-force type of attack, where the attacker simply runs the ISO defined set of commands to see if the card replies to one or more commands that it should not answer to.

151 As an undocumented command search can be highly standardized and automated, it should not take much more time than one day. Once all variations of Class, Instruction, Parameter 1 and Parameter 2 are tried and the answers recorded, the attacker analyses if there is any interesting attack mount point. Once an interesting answer has been determined the attacker builds a script to exploit the vulnerability. This could also be done by source code checking.

152 Whether the undocumented command may present attack points depends on the quality of the software (the separation of execution domains) and the type of command that is discovered.

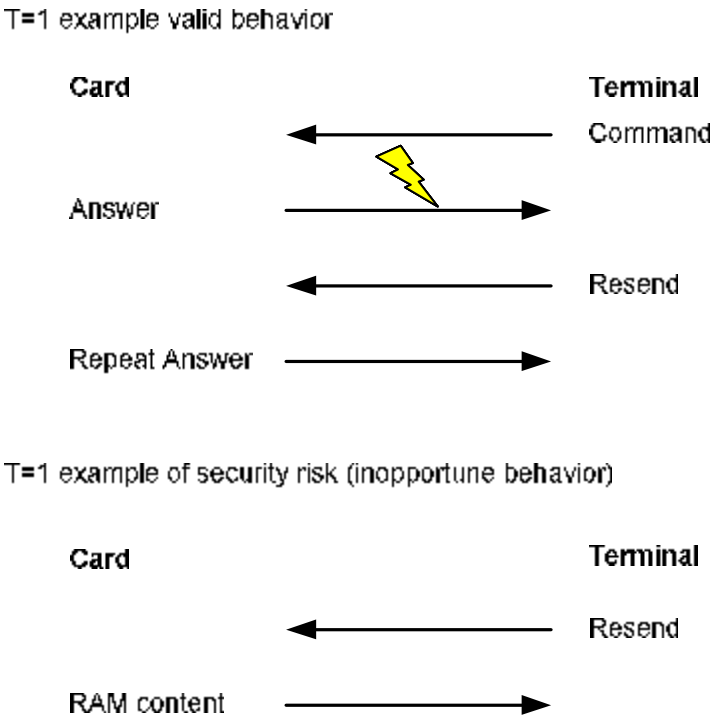
4.13 Editing commands

153 Editing commands is an attack step where the attacker tries to modify commands during the communication sequence to see if the card gives an unexpected reply (these commands may be in an interface specification, or they may have been discovered by observing message sequences or a command search as described above). These attack steps may enable vulnerabilities to be discovered and exploited (e.g. editing previously observed messages to supply a parameter that is too long may enable a buffer overflow attack). They may also expose timing differences that assist in reverse engineering of the software.

154 According to the security mechanisms associated to the API and the type of message, it may be easy or complex to forge a message (Mutual authentication, Secure channel, MAC, Ciphering, session key,...). However, as noted earlier, if an attack of this sort can be found then it will generally cause a TOE to fail evaluation.

4.14 Direct protocol attacks

155 A typical protocol attack is to try to send commands that the smartcard does not expect in its current state. For example: the ISO 7186-3 and 14443 protocols for smartcards contain a command for handling failure in the communication. Instead of starting a genuine communication, by sending this command an attacker may receive an un-initialized buffer, or the last buffer that was written. This example is shown in the following pictures.



156 Whether the TOE actually dumps the memory contents depends on the proper initialisation of I/O buffer pointer and length. The memory shown in the example might contain residual secret data, for example a DES session key that was just calculated. Therefore this attack may allow an attacker to retrieve secret data from the TOE.

4.15 Man-in-the-middle attacks

157 In this attack, the attacker hides in the communication path between two entities that are executing a valid communication. The attacker presents himself to either party as the other (valid) party. Some applications of Man in the middle attacks in public literature may be found in the following papers:

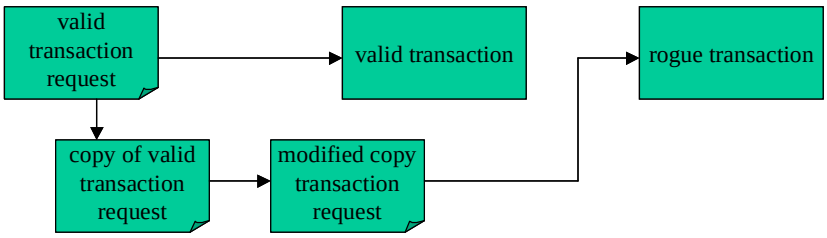
- An Example of a Man-in-the-middle Attack Against Server Authenticated SSL-sessions, Mattias Eriksson
- Man-in-the-Middle in Tunnelled Authentication Protocols, N. Asokan, Valtteri Niemi, Kaisa Nyberg, Nokia Research Center, Finland
- Why Cryptosystems Fail, Ross Anderson

4.16 Replay attacks

158 Replay attacks are possible when a mechanism does not check that a command is a genuine part of the current message sequence, or that a complete message sequence has not been used before (in general, a secure protocol should prevent this sort of attack by design[4]). An attacker uses a protocol analyser to monitor and copy packets as they flow between smartcard and reader or host. The packets are captured, filtered and analysed for interesting information like digital signatures and authentication codes. Once these

[4] Even where a protocol is designed to be secure, it may be possible to use a replay attack if a further attack step (such as a perturbation) is used to avoid a check that would otherwise detect and reject the replayed commands.

packets have been extracted, the packets are sent again (replayed), thus giving the attacker the possibility to get unauthorized access to resources.



159 The picture shows a situation where the attacker copies a valid transaction request, modifies it and sends a second request using the same (or slightly modified) versions of the messages. In general this type of attack might allow the attacker to get unauthorised access to a user's assets, for example a bank withdrawal or access to protected system resources.

160 The attack may be a full attack path, such as if a bank account withdrawal succeeds. In the case where system resources are accessed, it might be a partial attack path, depending on the nature of the resources that are accessed (e.g. the attacker is now able to communicate as an ordinary user and tries to get elevated privileges).

161 The replay attack might be countered by using sequence numbers with appropriate integrity protection, making the use of recorded valid messages much harder.

4.17 Bypass authentication or access control

162 This type of attack aims at getting unauthorised access to data residing on the smartcard respectively at performing operations which do not match the current life cycle state of processed data objects or of the Operating System. In particular, unauthorised reading or modification of personalisation data stored on the card, or a further (unauthorised) initialisation or personalisation of the product could be the target of such an attack scenario. This type of attack (which may also be whole program sequences) makes use of weaknesses in software implementation and is performed by a logical or physical attack on the Operating System and its processed data. The tools used are protocol attacks, either e.g. man-in-the-middle, replay, command editing or using commands which are undefined or not allowed in the current life cycle state of the Operating System. Furthermore, logical and physical attacks manipulating the program flow, status information (as the life cycle state of objects and of the Operating System) and access rules for objects processed by the Operating System have to be taken into account.

4.17.1 Description of Attack

163 This type of attack aims to get unauthorised access to data residing on the smartcard respectively to perform operations, which do not match the current life cycle state of processed data objects or of the Operating System. As an example, such an attack aims to read or modify personalised data that reside on the card or targets to perform a further (unauthorised) initialisation or personalisation of the product.

164 Getting unauthorised access to data stored on the smartcard can be obtained by various techniques:

- Impersonating the other side of the communication (known as 'man-in-the-middle'),
- using timing differences (by capturing and replaying commands),
- trying command variations (either editing valid commands or
- finding undefined commands),
- manipulation of access rules themselves,
- circumvention or manipulation of the request and evaluation of access rules during program execution.

165 Executing commands that are not allowed in the current life cycle state of the Operating System or of a data object can be as well obtained by various techniques:

- manipulation of the current life cycle state itself,
- circumvention or manipulation of the request and
- evaluation of the current life cycle state during program execution, and
- trying command variations (either editing valid commands or finding undefined commands).

4.17.2 Effect of Attack

166 The effect of the attack is unauthorised access to data residing on the smartcard respectively the possibility to perform operations, which do not match the current life cycle state of the Operating System or of data objects processed by the Operating System. In particular, such an attack could lead to the disclosure of stored secret data or to a further (unauthorised) initialisation or personalisation of the product. The attacker knows the attack has succeeded by analyzing the answers the smartcard gives during the (following) communication.

167 In general, the described attack scenario aims at the manipulation of the intended security structure integrated in the Operating System, in the applications set up on this Operating System and in the (application) data processed by the Operating System. The integrated access control to data objects and commands is affected.

168 Replay attacks have existed for a long time. Years ago, replay attacks were aimed at stealing passwords. Given the encryption strength of passwords these days, the focus of this type of attack has shifted to stealing digital signatures and keys.

169 The command editing attack aims to find commands that are not documented or using valid commands in a way that breaks the communication mechanisms in the TOE. The attacker may try to find improper bounds checking by sending longer commands than the TOE expects. He may try to send commands with unexpected values, forcing the smartcard to dump memory contents.

170 The manipulation of life cycle state information and access rules themselves, and the manipulation of their request and evaluation can be considered as a direct attack on the access control implemented in the Operating System and the applications running on this platform. In particular, the access control is modified or completely switched off in

a way that unauthorised access to secured data or the execution of not allowed commands is possible.

4.17.3 Characteristics of the Attack

171 The manipulations of life cycle state information and access rules require a physical attack on the smartcard and its Operating System and applications. The circumvention and manipulation of the request and evaluation of life cycle state information and access rules bases on a manipulation of the intended program flow what may be achieved by logical or physical means. An active countermeasure for securing life cycle state information and access rules and their request and evaluation during program execution could be to attach an integrity attribute and to check this attribute appropriately during program execution. More details concerning the characteristics of these attacks and effective countermeasures can be found in the sections 4.1 “Physical Attacks” and 4.3 “Perturbation Attacks”.

172 The attacks of logical nature as man-in-the-middle attacks, replay attacks, command editing are considered in detail in the sections Software Attacks 4.11.

4.18 Buffer overflow or stack overflow

173 This attack is applicable to open platforms.

174 Open platforms are defined in this document as smart card operating systems with the capability of running and downloading multiple applications.

175 Open platforms provide to the applications a set of services, in particular services to protect their sensitive data against external applications (unauthorized access and unexpected modification).

176 This attack could be performed through buffer overflow or stack overflow, produced by the execution of a malicious application.

177 Overflow, when not checked by the platform, can have various effects, such as overwriting existing content in the current stack.

178 The expected effect by the attacker here is the malicious application modifies the current execution context and switch to system privileges.

179 Gaining such privileges allow this application to virtually execute every operation and then disclose or modify secret data, e.g. modifying or disclosing the PIN of another application.

5 References

[AIS31] Functionality classes and evaluation methodology for physical random number generator, Version 1, 2001-09-25 and
the associated technical document: “A proposal for: Functionality classes and evaluation methodology for true (physical) random number generator”, Version 3.1, 2001-09-25, W. Killmann (T-Systems), W. Schindler (BSI)

[CC] Common Criteria for Information Technology Security Evaluation, Version 2.3, August 2005.

[CEM] Common Methodology for Information Technology Security Evaluation (CEM), Version 2.3, August 2005.

[CC-IC] The Application of CC to Integrated Circuits, CCDB-2006-04-003.

[COMPO] ETR-lite for composition, CCDB-2006-04-005.