Joint Interpretation Library

# Application of Attack Potential to Smartcards

Version 2.9
January 2013

This page is intentionally left blank

## Table of contents

**1 Introduction ....... 5**

**2 Scope ....... 5**

**3 Foreword: Workload for AVA_VAN.5 evaluation ....... 5**

**4 Identification of Factors ....... 5**

4.1 How to compute an attack ....... 6

4.2 Elapsed Time ....... 7

4.3 Expertise ....... 8

4.4 Knowledge of TOE ....... 9

4.5 Access to TOE ....... 11

4.6 Equipment ....... 11

4.7 Tools ....... 12

4.8 Open Samples/Samples with known Secrets ....... 13

4.9 Final Table ....... 21

4.10 Range for CC v2 ....... 22

4.11 Range for CC v3 ....... 22

**5 Examples of attack methods ....... 23**

5.1 Physical Attacks ....... 23

5.2 Overcoming sensors and filters ....... 24

5.3 Perturbation Attacks ....... 24

5.4 Retrieving keys with DFA ....... 26

5.5 Side-channel Attacks – Non-invasive retrieving of secret data ....... 26

5.6 Exploitation of Test features ....... 28

5.7 Attacks on RNG ....... 28

5.8 Ill-formed Java Card applications ....... 30

5.9 Software Attacks ....... 30

5.10 Applet isolation ........................................................................... 39

# 1 Introduction

1 This document interprets the current version of Common Criteria Methodology [CEM] (annex A.8 for CC v2, annex B.4 for CC v3). This work has been based on smartcard CC evaluation experience and input from smartcard industry through the International Security Certification Initiative (ISCI) and the JIL Hardware Attacks Subgroup (JHAS).

2 This chapter provides guidance metrics to calculate the 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 the operational behaviour of a smartcard and not to applications specific only to hardware or software.

3 This document is compatible with CC v2 and CC v3 [CC].

# 2 Scope

4 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 Foreword: Workload for AVA_VAN.5 evaluation

5 No rigid rules can be given on how much time should be spent on a typical smartcard VAN.5 evaluation by a competent lab, but the following guidance shall none-the-less be provided in an effort to harmonise evaluations and the various national schemes alike: Assuming the CC vulnerability analysis has already been performed the evaluation testing from scratch for a new IC should take about 3 man months, depending on the complexity of the IC such as the number of cryptographic services, interfaces, etc. The total evaluation time for composite evaluations using a certified IC for VLA.4 / VAN.5 testing activities is of the order of 1-3 man months, depending on the complexity of the platform, such as open platform, native platform, number of APIs, etc.. It is possible to deviate from this guidance, but some reasoning will have to be provided to the CB.

6 It is an assumption of this interpretation that the Certification Bodies will ensure that there is harmonisation not only nationally, but also 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 points for the time or expertise to develop the attack (as discussed above).

# 4 Identification of Factors

7 Note about CC v3.1 :

8 With Common Criteria version 3.1, there is no more distinction between the identification phase and the exploitation phase but within the smartcard community, the risk management performed by the user of CC certificates clearly required to have a distinction between the cost of “identification” (demonstration of the attack) and the

cost of "exploitation" (e.g. once a script is published on the World Wide Web). Therefore, this distinction is kept when calculating the attack potential for smartcard evaluations. Although the distinction between identification and exploitation is essential for the smartcard evaluation to understand and document the attack path, the final sum of attack potential is calculated by adding the points of these two phases, as both phases together constitute the complete attack.

## 4.1 How to compute an attack

9 Attack path identification as well as 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, as well as whether or not open samples or samples with known secrets had been used. Even if the attack consists of several steps, identification and exploitation need only be computed for the entire attack path.

10 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 an 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.

11 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, no points for the development of the attack (in terms of time or expertise) will be used in the attack potential calculation any more.

12 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 a 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.[1] This means that the elapsed time,

[1] This assumption is the worst-case scenario: The information obtained in a first attack (in the Identification phase) is fully shared with other attackers who wish to exploit this attack (Exploitation phase). This assumption

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 and physical location 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 mere 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). As a general rule, no points can be awarded for the exploitation phase at all when, e.g., a secret master key common to all TOEs under investigation has been compromised in the identification phase. This is so as the script defining details to be passed on between the identification and exploitation phase will already contain the information on this master key. An example would be storing a master key in ROM.

13 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.

14 To complete an attack potential calculation the points for identification and exploitation have to be added as both phases together constitute the complete attack. 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.

## 4.2 Elapsed Time

15 Compared to the “Elapsed Time” factor as given in CEM, further granularity is introduced for smartcards. In particular, a distinction is drawn between one week and several weeks. The Elapsed Time is now 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 | * | * |

is not always correct, in particular when the attack happens for commercial profit and sharing would have to happen between rivaling criminal organisations.

**Table 1: Rating for Elapsed Time**

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

17 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.

18 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 here.

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

(a) Consider 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) Consider 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.

20 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 be to decide on the meaning of "Not practical" only in a specific attack scenario.

## 4.3 Expertise

21 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 level of knowledge of the TOE commensurate to that of the developer (e.g. knowledge of product standards and specifications).

22 The 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<br>• Algorithms<br>• Protocols<br>• Hardware structures<br>• Principles and concepts of security | Familiar with<br>• Developers knowledge namely algorithms, protocols, hardware structures, principles and concepts of security<br>and<br>• Techniques and tools for the |

| | Definition according to CEM | Detailed definition to be used in smartcard evaluations |
|---|---|---|
| | | definition of new attacks |
| b) Proficient | Familiar with<br>• security behaviour | Familiar with<br>• security behaviour, classical attacks |
| c) Laymen | No particular expertise | No particular expertise |

**Table 2: Definition of Expertise**

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

**Table 3: Extent of expertise**

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

24 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**

## 4.4 Knowledge of TOE

25 The CEM v.2.3 states that "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. In general it is expected that all knowledge required in the Exploitation phase will be passed on from the Identification phase by way of suitable scripts describing the attack.

26 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.

27 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 cannot be gained in another way.

28 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 hand by taking into account two types of information (HLD/LLD and Implementation Level). (See [CC-IC])

29 It may occur that for sophisticated attacks, several types of knowledge are required. In such cases, the highest 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**

## 4.5 Access to TOE

30 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“).

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

- the attack succeeds only with some probability on a given device such that a number of devices need to be tried out,
- the attack succeeds only after having destroyed a number of devices (on average),
- the attacker needs to collect information from several copies of the TOE.

32 In this case, TOE access is taken into account using the following rating:

| | Identification | Exploitation |
|---|---|---|
| < 10 samples | 0 | 0 |
| < 30 samples | 1 | 2 |
| < 100 samples | 2 | 4 |
| > 100 samples | 3 | 6 |
| Not practical | * | * |

**Table 6: Rating for Access to TOE**

33 “Not Practical” is explained as follows:

- 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.

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

## 4.6 Equipment

35 In order to clarify the 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.

36 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 equipments. 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.

37 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 due to new forms of attacks

38 It may happen that for sophisticated attacks several types of equipment are required. In such cases by default the highest of the different equipment factors is chosen.

## 4.7 Tools

39 The border between standard, specialized and bespoke cannot 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 |
| Flash light | Standard |
| Low-end visible-light microscope | Standard |
| Climate chamber | Standard |
| Voltage supply | Standard |
| Analogue oscilloscope | Standard |
| Chip card reader | Standard |
| PC or work station | Standard |
| Signal analysis software | Standard |
| Signal generation software | Standard |
| High-end visible-light microscope and camera | Specialized |
| UV light microscope and camera | Specialized |
| Micro-probe Workstation | Specialized |
| Laser equipment | Specialized |
| Signal and function processor | Specialized |
| High-end digital oscilloscope | Specialized |
| Signal analyzer | Specialized |
| Tools for chemical etching (wet) | Specialized |
| Tools for chemical etching (plasma) | Specialized |

| Tool | Equipment |
|---|---|
| Tools for grinding | Specialized |

**Table 7: Categorisation of Tools (1)**

### 4.7.1 Design verification and failure analysis tools

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

41 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 still, 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)**

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

43 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. Testbenches for side-channel and fault attacks are normally considered to be too similar and not different enough.

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

## 4.8 Open Samples/Samples with known Secrets

### 4.8.1 Purpose of this section

45 In a composite evaluation as a rule, the properties of the hardware are taken from the information supplied with the documentation from the certification of the underlying

platform IC. For this purpose, the CC supporting document [COMPO] specifies the process, called “composite smartcard evaluation”.

46 In general, the ETR-FOR-COMPOSITION should be written so as to contain enough information to evaluate and certify a composite product. In certain cases, it might be opportune to use “open samples” to speed up the evaluation process. The use of the “open samples” or “samples with known secrets”, its scope, and the implications on the evaluation and the attack rating is described in this section.

### 4.8.2 Definition of “open samples / Samples with known Secrets”

47 Within the context of a composite evaluation, the term “open samples” stands for samples were the evaluator can put SW on the HW platform at his own discretion that bypasses countermeasures prescribed in the IC guidance. The intention is to use test SW without SW countermeasures but not deactivate any IC inherent countermeasures. In addition, another possibility is to enable the evaluator to define one or more pieces of secret data, such as a PIN or key, where this ability would not be available under the normal operation of the TOE. The SW should serve to highlight IC properties described in the IC ETR-FOR-COMPOSITION considering the special use of the HW in the TOE but not be used to repeat the IC evaluation. If the IC allows different configurations, the configuration implemented in the TOE shall be used. With these samples, it is thus possible to characterise the HW without SW.

48 “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 a PIN or key for performing either passive (monitoring) or fault attacks, yet without deactivating any countermeasures.

### 4.8.3 Use of “open samples / Samples with known Secrets”

49 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].

50 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, these documents will be a guidance on how the HW has to be used to achieve the security objectives.

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

52 In some 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 as well as the effectiveness of SW countermeasures.

53 Also, the platform may be used in a way that was not foreseen by the HW developer and the composite evaluator, or the SW provider may not have followed the

recommendations provided with the HW and implemented different countermeasures where the effectiveness is not yet proven.

54 Finally, 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.

55 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" and/or the use of "samples with known secrets".

56 As a rule, a composite evaluation should not require the use of "open samples". However, if an efficient and meaningful evaluation in a maintainable time is only possible with the use of "open samples", then certain rules should be followed:

- The purpose of open samples is to set up tests for the composite evaluation and not to repeat the hardware-evaluation.
- The use of open samples and the information flow between parties is discussed and agreed upon between the certification body, the evaluator, the developer of the composite TOE and the developer of the open samples.
- The time spent on the dedicated "open sample" tests is restricted to one or two weeks.
- The goal and type of the tests is discussed and made known to all parties as defined in the information flow agreement.
- Failures and observations resulting from the tests are communicated and made known at least to the certification body of the composite TOE. The certification body of the composite TOE shall take appropriate steps together with the certification body of the HW evaluation.
- The rating should make provision for the judgement whether or not the attack would have been possible without the use of "open samples".

### 4.8.4 Implications on the composite evaluation

57 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.

58 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.

59 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.

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

### 4.8.5 Implications on the composite rating

61 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.

### 4.8.6 Background of the use of “samples with known secret” to accelerate the evaluation

62 An additional possibility to accelerate the evaluation especially where cryptographic operations are involved is the use of “samples with known secret”. With these samples the evaluator knows the “secret” (key). This allows either comparing of retrieved data (e.g. as deduced from passive analysis) against the known “secret”, or it may be useful in a profiling step required for some attacks. The evaluator therefore has a simplified way to determine if his attack has revealed the correct secret. He can stop after retrieving parts of the “secret” and estimate the remaining time to find the complete “secret”.

63 However, a rating based on such samples has to be carefully considered because the attack might only be made possible by the availability of “samples with known secret”.

64 For instance:

- To extract the complete key might prove to be very time consuming. With some error in the retrieved key and no possibility to decide which part of the secret is not correct an attack might not be possible.
- A profiling stage is sometimes required to perform some attacks, such as template attacks. Knowing the key, and then the intermediate values of the algorithms, may then make an attack possible.

65 In general the rating of “samples with known secret” is comparable to the rating of “open samples”. Therefore, both tools are combined here.

### 4.8.7 Calculating the attack potential

66 As with other aspects of an attack, the evaluator has to estimate the value of the factors (time, access to TOE, etc) for an attacker.

67 Where open samples exist, collusion (or direct attack, such as theft) to obtain them is possible in the same way that the evaluation takes into account a possible collusion or direct attack for an attacker to get design information (down to the implementation level).

68 A factor “open sample” is therefore defined in the attack potential table, with points in the identification phase for “open samples” used during evaluations. The same factor should be applied for the “samples with known secret”.

69 When rating an attack that makes use of open samples / samples with known secrets, the evaluator must first determine (at least theoretically) and describe the way in which an attacker could carry out the attack on the real TOE (instead of on the open sample / sample with known secret). Having determined this, the evaluator will perform two calculations, using open samples / samples with known secret:

- Estimating the value for each factor for an attacker without access to open samples / samples with known secrets.
- Giving the values for each factor corresponding to what he has done (had he completed the entire attack):
  - Time spent, destroyed samples, Expertise, Knowledge of the TOE, equipment
  - Adding the points corresponding to the open samples used

70 Should it turn out that the attack is not practical when not using open samples or samples with known secrets, then that rating has be used and the open samples / samples with know secrets rating discarded. In all other cases 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.

71 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 life cycle.

72 For ICs, the protection level will be analysed during the IC evaluation and stated in the ETR-FOR-COMPOSITION.

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

74 Because of the similarity in the threat to the TOE, the rating for open samples (with and without known secrets) should be defined according to the values of the Knowledge of the TOE factor: PUBLIC, RESTRICTED, SENSITIVE and CRITICAL:

- PUBLIC:
  - Open samples: 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).
    - Samples with known secrets: This concerns secrets easily deducible from information already rated in “knowledge of the TOE”.
- RESTRICTED:
    - Open samples: 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.
    - Samples with known secrets: Typically applies to secrets where a specific decision and action is required to release the information (so it is not, for example, automatically available for anonymous access via a website), and where the recipient is made aware that the data is potentially useful to an attacker (e.g. via guidance information). In some cases it may be possible for an attacker to find out or deduce the information, but its availability still provides convenience (and perhaps a saving of time).
- SENSITIVE:
    - Open samples: Protected as the HLD/LLD design levels are.
    - Samples with known secrets: Secrets are only shared by a limited number of clearly defined and identified people or devices, with strong access controls. Handling of the Secret data is governed by specific and appropriate written procedures to protect it, and there is a clear method by which the Secret data is identified as requiring these procedures (e.g. by labelling the data).
- CRITICAL:
    - Open samples: 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.
    - Samples with known secrets: Secrets were generated inside the sample and are only owned by it, or in another module which does not make these secrets available outside the module (except to the sample). These secrets are therefore not available outside the card, and possibly the module, under normal conditions. Only under exceptional conditions could these secrets be known, for example by providing the evaluator with either specific commands to access the secrets (not available in any normal configuration of the TOE), or special samples with static secrets instead of dynamic secrets (fixed in personalization phase for instance). As with Open Samples at the Critical level, this requires that there are very few Open Samples produced, that they have very strong control over authorisation for their release and delivery to the recipient,

and to have assurance that the receiving organisation will control the samples so as to provide equivalent limits on their availability.

75 The composite evaluation has also to define if the use of “open samples” **and** “samples with known secret” accumulates the efforts in 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.

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

| **(Identification phase only)** | **PUBLIC or not required** | **RESTRICTED** | **SENSITIVE** | **CRITICAL** |
|---|---|---|---|---|
| Open Samples | 0 | 2 | 4 | 6 |
| Samples with known secret | 0 | 2 | 4 | 6 |

### 4.8.8 Impact on the evaluation of an IC with guidance

77 As such, the concept of “samples with known secrets” does not apply to HW IC evaluations, since the final application is not known at this point in time. For instance, open, unprotected applications / APDUs residing alongside the secured ones may effectively allow to perform an analysis equivalent to using “samples with known secrets” in the first place.

78 The situation is more complex with regards to the concept of “open samples”. Here it is useful to distinguish two different classes of countermeasures that may be described in the IC guidance.

- Simple countermeasures are those that are effectively equivalent to a switch. For instance, the IC guidance may require that the TOE shall only be used with internal clock, or only with a clock-skipping mode enabled. In those cases it may be advantageous for the IC evaluator to switch these features off and thereby speed up the evaluation. The assessment will then involve generating two different ratings, one with an estimate for the time that would have been spent on the attack had the countermeasure been enabled, and a second rating along the rules for open samples. As before, in the end the minimum of the two ratings will be chosen.
- Complex countermeasures are those that require more or less complex SW code to be generated in the final application where it can be expected that some variability will exist from one implementation to another. Typical examples here are countermeasures against fault attacks. In such a case the concept of open samples does not apply to HW IC evaluations, since there is no way of knowing whether a final product based on this IC does implement all SW countermeasures recommended in the HW IC guidance in such a way that no loophole could possibly exist.

### 4.8.9 Impact on the evaluation of a firmware / crypto library of an IC with guidance

79 Crypto libraries and other supporting routines for a HW IC that are evaluated within the composite evaluation scheme (often separately from the HW IC evaluation) are somewhat special in that they are not a final product in their own right, but rather are to be used in one or more final products, which in turn may themselves be subject to a composite evaluation.

80 The concept of “samples with known secrets” usually does not apply here since the crypto library has in general no control over the handling of those secrets outside its boundaries. This is different, though, for secrets generated and maintained within the crypto library that are not exported.

81 However, the concept of “open samples” may apply more often, depending on the circumstances. A typical example would be countermeasures against light attacks or SPA-DPA attacks that are applied under the control of the crypto library. Provided these countermeasures cannot be switched off in the final product, the crypto library may be considered to be equivalent to a final product in this respect, and consequently the concept of “open samples” of the composite evaluation scheme applies. It does not matter here whether these are countermeasures that have been actually suggested in the HW IC guidance or not.

## 4.9 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 |
| < 30 samples | 1 | 2 |
| < 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 |
| Restricted | 2 | NA |
| Sensitive | 4 | NA |
| Critical | 6 | NA |

**Table 10: Final table for the rating factors**

82 (1) If clearly different testbenches consisting of specialised equipment are required for distinct steps of an attack this shall be rated as bespoke. Testbenches for side-channel and fault attacks are normally considered to be too similar and not different enough.

83 * 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.

## 4.10 Range for CC v2

84 The following table replaces table A.8 of CEM, para 1835 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 for CC v2**

85 *final attack potential = identification + exploitation.

## 4.11 Range for CC v3

86 The following table replaces table B.4 of CEM, para 1869 for smartcards.

| Range of values* | TOE resistant to attackers with attack potential of: |
|---|---|
| 0-15 | No rating |
| 16-20 | Basic |
| 21-24 | Enhanced-Basic |
| 25-30 | Moderate |
| 31 and above | High |

**Table 11: Rating of vulnerabilites for CC v3**

87 *final attack potential = identification + exploitation.

# 5 Examples of attack methods

88 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).

89 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.

90 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, nor should this catalogue of attacks be considered as an exhaustive list. On the contrary, the manufacturers and labs are encouraged to search for new attacks and attack variants as part of their evaluation activities.. For each TOE the evaluation lab conducting the evaluation will 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 are likely also required.

91 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.

## 5.1 Physical Attacks

92 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.

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

94 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).

95 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

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

## 5.2 Overcoming sensors and filters

97 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.

98 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.

99 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.

100 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)

101 The main impacts are:

102 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).

103 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

## 5.3 Perturbation Attacks

104 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 operating the IC outside its intended operating environment (usually characterised in terms of temperature, Vcc and the externally supplied clock frequency) or by

applying one or more external sources of energy during the operation of the IC. These energy sources can be applied at different times and/or places on the IC.

105 The attack will typically aim at reducing the strength of cryptographic operations by creating faults that can be used to recover keys or plaintext, or to change the results of checks such as authentication or lifecycle state checks, or to change the program flow.

106 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.

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

108 The main impacts are:

109 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.
- Modifying a value that is stored in volatile memory. The modified value is effective until it is overwritten by a new value, and could therefore be used to influence the processing results.
- Changing the characteristics of random numbers generated (e.g. forcing RNG output to be all 1’s) – see Attacks on RNG 128 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
    - Replacing an instruction with another (benign) one
    - Inverting a test
    - Generating a jump
    - Generating calculation errors

110 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).

111 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.

## 5.4 Retrieving keys with DFA

112 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.

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

- Search for a suitable single or multiple 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

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

115 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.

116 The main impacts are:

117 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.

118 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.

## 5.5 Side-channel Attacks – Non-invasive retrieving of secret data

1 Side-channel attacks target secret information leaked through unintentional channels in a concrete, i.e. physical, implementation of an algorithm. These channels are linked to physical effects such as timing characteristics, power consumption, or electromagnetic radiation.

119 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.

120 When an IC is operating, each individual element will emit electromagnetic radiation in the same way as any other conductor with an electrical current flowing through it. Thus, as this current varies with the data being processed, so does the electromagnetic radiation emitted by the TOE. Electromagnetic Analysis (EMA) attacks target this variant of information leakage. These attacks are sometimes referred to as SEMA (Simple Electromagnetic Analysis), or DEMA (Differential Electromagnetic Analysis). They 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).

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

122 For the sake of unity in what follows SPA and DPA will denote not only attacks based on measurements of the power consumption, but are understood to cover their "cousins" in electromagnetic attacks as well, unless stated otherwise.

123 Implementations that include countermeasures like Boolean masking that resist first order DPA may be vulnerable to higher-order DPA. This attack requires that the attacker is able to correlate more than one data point per TOE computation using hypotheses on intermediate states that depend on secret key parts. Generally, the effort for a higher-order DPA is higher than for first-order DPA, particularly during the Identification phase. This is partly also because a higher-order DPA needs to be tailored to the countermeasures in place.

124 The outcome of the attack may be as simple as a characteristic trigger point for launching other attacks (such as a general perturbation or a 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.

125 The main impacts are:

126 It lies in the very nature of side-channel 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 (e.g., for retrieving the PIN), 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. EMA may be deployed to localise, say, the DES coprocessor on the physical layout of the chip, which in turn can then be used to launch other attacks.

127 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.

## 5.6 Exploitation of Test features

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

129 If an attacker is able to circumvent the protection of the test features he can use the test interface and test functions as appropriate for the intended attacks. These further attacks might lead to disclosure or corruption of memory content, but this depends on the possibilities of the test mode and has to be considered case by case.

130 The typicalimpacts of a successful attack are:

- 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.
- The attacker is able to re-configure the life cycle data or error counters using a test function. Thereby an attacker is able to continue his analysis on the same device, even when a lifecycle status change would otherwise have stopped him.

## 5.7 Attacks on RNG

131 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) .

132 A RNG considered here can be one of the following types[2]:

- 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),

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

- hybrid RNG (HRNG), which consists of a TRNG and a PRNG with a variety of state update schemes,

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

134 A true RNG may be attacked by[3]:

- 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)
- 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)

135 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)

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

137 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.

138 The main impacts are:

139 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.

140 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

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

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.

141 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.

## 5.8 Ill-formed Java Card applications

142 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.

143 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.

144 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.

145 The main impacts are:

146 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).

## 5.9 Software Attacks

147 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.

148 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).

149 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.

150 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.

151 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

152 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.

153 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)

154 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.

### 5.9.1 Information gathering

#### 5.9.1.1 Introduction

155 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.

156 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.

157 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[4]). The information gathered is analysed to see whether it can be used to mount an attack to 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.

#### 5.9.1.2 Attack Step Descriptions

Observing Message Sequences

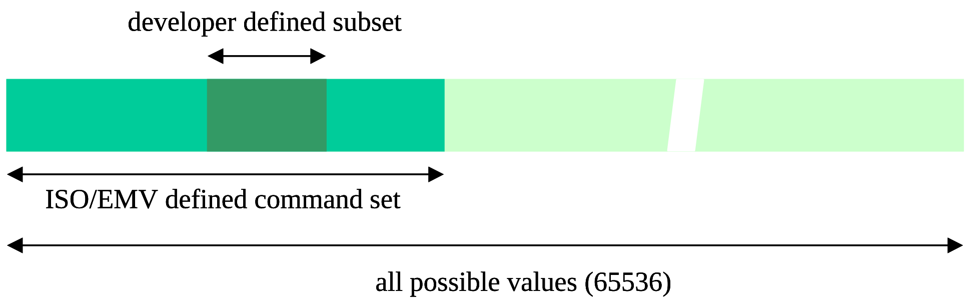
158 Observing message sequences may result in:

- 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

159 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.

[4] 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.



160 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

161 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.

162 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.

163 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.

### 5.9.2 Editing commands

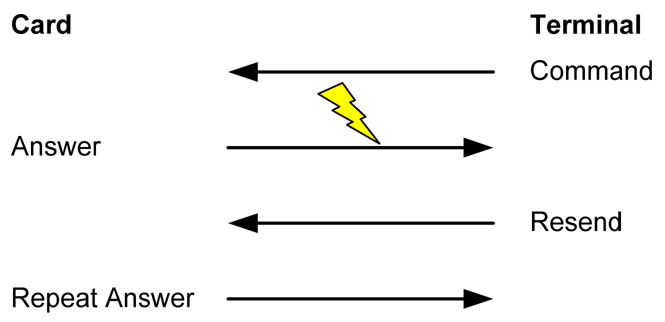
164 Editing commands is an attack step where the attacker tries to modify commands during the communication sequence to see if the card gives an unexpected reply (these commands may be in an interface specification, or they may have been discovered by observing message sequences or a command search as 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.

165 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.

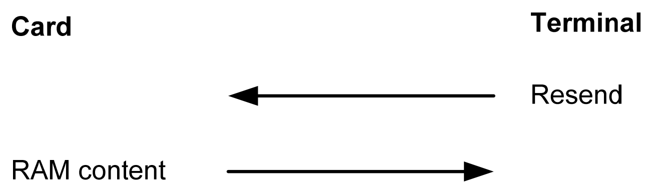
### 5.9.3 Direct protocol attacks

166 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.

T=1 example valid behavior



T=1 example of security risk (inopportune behavior)



167 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.

### 5.9.4 Man-in-the-middle attacks

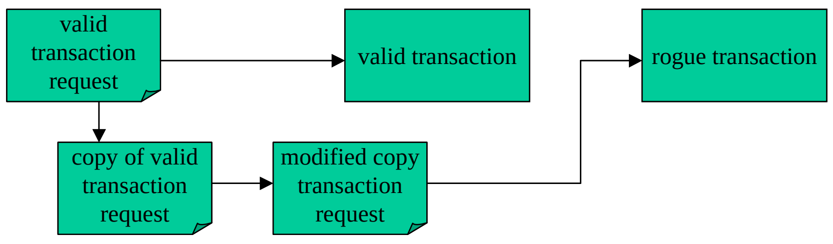
168 In this attack, the attacker hides in the communication path between two entities that are executing a valid communication. The attacker presents himself to either party as the other (valid) party. Some applications of Man in the middle attacks in public literature may be found in the following papers:

- 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

### 5.9.5 Replay attacks

169 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[5]). 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 packets have been extracted, the packets are sent again (replayed), thus giving the attacker the possibility to get unauthorized access to resources.


valid transaction request
valid transaction
rogue transaction
copy of valid transaction request
modified copy transaction request

170 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.

171 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).

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

### 5.9.6 Bypass authentication or access control

173 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

[5] 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.

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.

#### 5.9.6.1 Description of Attack

174 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.

175 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.

176 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).

#### 5.9.6.2 Effect of Attack

177 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.

178 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.

179 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.

180 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.

181 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.

#### 5.9.6.3 Characteristics of the Attack

182 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 5.1 “Physical Attacks” and 5.3 “Perturbation Attacks”.

183 The attacks of logical nature as man-in-the-middle attacks, replay attacks, command editing are considered in detail in the sections Software Attacks 5.9.

### 5.9.7 Buffer overflow or stack overflow

184 This attack is applicable to open platforms.

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

186 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).

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

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

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

190 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.

## 5.10 Applet isolation

### 5.10.1 Multi-application platforms security stakes

A multi-application platform describes a set of hardware and software built with the aim to run more than one application at the same time.

The assets that may need to be protected in a multi-application environment are:

- Loaded application data (including keys).
- Loaded application code.
- Underlying platform data.

Applet isolation is the target of various types of attack techniques to reach these assets.

### 5.10.2 Partial attacks

There is an existing set of technologies applicable to ensure the isolation of applications. These technologies are usually specified in standards, and can be combined in smart card devices.
When performing a full attack, an attacker may need to defeat one or a combination of these technologies. The term partial attacks is used here to describe attacks that have to be combined in the performance of a full attack.

#### 5.10.2.1 GlobalPlatform partial attacks

##### 5.10.2.1.1 GlobalPlatform principle

The GP standard comes with the definition of a framework for application interoperability and management. This framework is specified by the GP specification and we can identify the main components as follows:

- Open GlobalPlatform Environment (OPEN)
  OPEN is responsible for command dispatching, (optional) multiple logical channel management, management of application and card lifecycle.
- Security domain (SD)
  A security domain represents a smart card actor on the card. It provides common security services for applications which are associated to it e.g. various kinds of cryptographic services, secure messaging as well as application personalization.
- Cardholder verification methods
  In particular, this gives the possibility for a unique user PIN number to be used by all applications.

#### 5.10.2.2 Description of a partial attack example

191 The aim of attacking GlobalPlatform is to allow an attacker to illegally load an application onto the TOE, *i.e.*, without knowing the loading keys values.

The attack is not performed on the cryptographic computations involved in the GlobalPlatform mutual authentication process and subsequent secure messaging commands.

The attack is performed on the code execution of the security domain with content management privilege. The attack exploits here a potential vulnerability in the robustness of the code execution flow against perturbation attacks. The idea is here to force the execution of any content management APDU command (`INSTALL [for load]`, `LOAD`, etc) whereas no secure channel has been opened.

### 5.10.3 On-card and off-card bytecode verifier partial attacks

#### 5.10.3.1 On-card and off-card bytecode verifier principle

These two kinds of byte code verifiers have a different behaviour:

- The On-Card Verifier performs its checks during the applet installation phase (link operation), at which point it can perform a structural verification of the CAP file and type verification.
- The Off-Card Verifier induces different organizational issues, as there is a need to guarantee that the application loaded on the card was actually checked by the off-card verifier. Off card verifier performs a structural analysis of the CAP file and a type verification (it simulates the execution of byte code).

#### 5.10.3.2 Description of a partial attack example

Basic type confusion attacks modify the reference of an object by the reference of another object. For instance, we can assign the address of a byte array to a short array in order to dump memory located after the byte array. There are two examples of attacks based on type confusing:

- Create a type confusion not detected by an On-Card Verifier enabling us to dump and modify a part of the memory content.
- Using a well-formed CAP file abusing the transaction mechanism in order to create a type confusion.

192

### 5.10.4 Defensive virtal machine partial attacks

#### 5.10.4.1 Defensive virtual machine principle

##### 5.10.4.1.1 Semi-defensive virtual machine

The semi-defensive virtual machine prevents type confusion by disallowing certain byte code execution sequences. Both virtual machines with off-card and on-card byte code verifiers are considered semi-defensive virtual machines.

##### 5.10.4.1.2 Defensive virtual machine

The defensive virtual machine[6] can analyze the byte code dynamically during the APDU execution (ex: type verification and structural verification) and does not require off- or on-card byte code analysis to prevent type confusion.

[6] There is no longer any definition of the defensive virtual machine in the version 2.6 of the Java Card system protection profile,

#### 5.10.4.2 Description of example of a partial attacks

193 The goal of an attack on a defensive virtual machine is to trick the virtual machine in allowing types to be confused. Such an attack may be possible when the defensive virtual machine is implemented only partially.

An ill-formed applet containing byte codes in illegal order is loaded onto the target which then, when defensive checks are not present or incomplete, causes a type confusion. This type confusion can then possibly be used to read persistent and transient data of the JCRE and other contexts not belonging to attacker's context.

194 A fully fledged type confusion attack uses the type confusion attack itself, the knowledge of the virtual machine meta data, and its application in a single attack applet able to read or write persistent and transient memory.

195

### 5.10.5 Firewall partial attacks

#### 5.10.5.1 Firewall Principle

The Java Card firewall limits access to object references by their context. Only objects created within the same context can be referenced. Access to resources outside the context of an object is possible through the Java Card Firewall by means of the Shareable Interface Object mechanism. Static members are excluded from firewall control and their accessibility does not depend on contexts.

#### 5.10.5.2 Description of partial attacks

196 Malicious applets in the Java Card environment could be used to challenge the restrictions imposed by the Java Card Firewall by attacking the context switching mechanisms. These malicious applets are well-formed and do pass byte-code verification. This attack may be easier to mount then ill-formed applet attacks as a malicious applet attack cannot be detected by byte code verification. On the other hand, this attack can only succeed if the firewall of the TOE is flawed.

### 5.10.6 Multos partial attacks

#### 5.10.6.1 Multos principle

MULTOS implements the following countermeasures:

1. Instructions, primitives and APDU commands do not allow addresses manipulation.
2. The Firewall: applet isolation, code space and data space isolation (for instance, we can't perform a jump from code to data).
   It is not possible to manipulate components contrary to Java Card (for instance in order to forge an address by deleting an element in the Reference location).
3. The MULTOS Application Abstract Machine provides each application with its own memory space.
4. The Loaded application can be encrypted

#### 5.10.6.2 Description of an example of partial attacks

This attack is a combined attack. Its aim is to attempt to read a block of data with an invalid size (a great one) and to perform a fault injection in order to bypass the firewall.

The firewall ensures that an application cannot access to another application space. If the attacker tries to execute an instruction which attempt to read a block of data with an invalid block length, the firewall will detect that the current application attempts to access to other application space and so will return an error. The evaluator needs to perform a fault injection in order to bypass this check and so succeeding to dump a part of memory.

### 5.10.7 Full attack path

197 The full attack paths combines partial attacks to get illegally access to sensitive resources (for example PINs and keys).

This attack is the combination of:

1. Getting a memory dump to locate assets and/or sensitive code through physical attacks or software attacks
2. Loading a malicious applet through through a partial attack on GlobalPlatform.
3. Type confusion to manipulate the objects identified in step 2 with the malicious applet through attacks on bytecode verifier or attacks ondefensive virtual machines.
   The attacker is able in the malicious applet to illegally manipulate a memory address of an object of another context. In this description, this is achieved through type confusions attacks.
4. Attack on the firewall using physical perturbations to execute the getKey method on the object or to execute an arbitrary code on an object of a different context.
   The attacker uses physical perturbations to bypass Java Card/Multos Firewall restrictions while manipulating objects out of the legitimate bounds.

Step 1 and step 2 are used to calibrate the attack. Step 3 and 4 are detailed here because in the partial attacks described in the previous chapters, we assume that a single malicious applet can perform every operation whereas in more realistic examples, a malicious applet can only handle its own objects. That's why here a perturbation is used to bypass the firewall restriction.

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.

Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 2, september 2007.

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

Common Methodology for Information Technology Security Evaluation (CEM), Version 3.1, Revision 2, september 2007.

[CC-IC] The Application of CC to Integrated Circuits,
Version 3.0, February 2009, Joint Interpretation Library
Version 3.0, Rev.1,March 2009, CC Supporting Document CCDB-2009-03-001.

[COMPO] Composite product evaluation for Smartcards and similar devices
Version 1.0, Septembar 2007, Joint Interpretation Library
Version 1.0, Rev.1, September 2007, CC Supporting Document ,CCDB-2007-09-001.