Joint Interpretation Library

Application of Attack Potential to Smartcards and Similar Devices

Version 3.0
April 2019

This page is intentionally left blank

Table of contents

1 Introduction

1 This document interprets the current version of Common Criteria Methodology [CEM] (annex B.4). 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 or similar device and not to applications specific only to hardware or software.

2 Scope

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

3 Foreword: Workload for AVA_VAN.5 evaluation

4 No rigid rules can be given on how much time should be spent on a typical smartcard or similar device VAN.5 evaluation by a competent lab, but the following guidance shall nonetheless 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 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 Certification Body.

5 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 below).

4 Identification of Factors

6 In the Common Criteria there is no distinction between the identification phase and the exploitation phase of an attack. However, within the smartcard community, the risk management performed by the user of CC certificates clearly requires 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 Internet). Therefore, this distinction must be made when calculating the attack potential for smartcard or similar device evaluations. Although the distinction between identification and exploitation is essential for the 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

7 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.

8 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.

9 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 in time determined by the relevant Certification Body and then harmonized by JHAS. Once this point in time has been determined, no additional points for the development of the attack (in terms of time or expertise) will be used in the attack potential calculation any more.

10 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, 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

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

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 a consequence 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 and the ROM content has been read out, decrypted or descrambled during the identification phase.

11 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.

12 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

13 Compared to the “Elapsed Time” factor as given in [CEM], further granularity is introduced for smartcards and similar devices. 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 (see below)	*	*

Table 1: Rating for Elapsed Time

14 If an attack path has been identified and there are well-understood analysis results that allow to extrapolate the elapsed time for the actual security configuration of the TOE, then Table 1 shall be extended by Table 1a:

	Identification	Exploitation
> four months	6	10

Table 1a: Extra rating step for Elapsed Time

15 It is not reasonable to expect an evaluation lab to spend extra time on the attacks. Hence, Table 1a only applies as a reasonable exception. And, this exception must be justified with analysis/measurement results enabling to define a rationale for a scalable time factor in the attack. This means that e.g. intuition and success by chance are not sufficient criteria.

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, which is 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 or similar device with comparable characteristics 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 or similar device as TOE used for an online system, where the TOE 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 if an attacker can extract the keys in one week.

(b) Consider a smartcard or similar device as TOE, 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 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 and similar devices, expertise levels are defined based on the attacker’s ability to define new attacks (hardware, software, cryptography) and to operate the necessary tools.

22 The expertise necessary to carry out an attack may cover several disciplines: chemical, ability to drive sophisticated tools, cryptography, etc.

	Definition according to [CEM]	**Detailed definition to be used in smartcard or similar device evaluations**
a) Experts	Familiar with • Implemented algorithms, protocols, hardware structures, security behaviour, principles and concepts of security employed and • Techniques and tools for the definition of new attacks, cryptography, classical attacks for the product type, attack methods, etc.	Identical to the [CEM], but at a level commensurate to experts in the field of smartcards or similar devices.
b) Proficient	Familiar with • security behaviour of the product type	Familiar with • security behaviour, familiar with laboratory measurements and equipment: can for example apply logical attacks, probing attacks and do non invasive analysis using laser, oscilloscope and similar.
c) Laymen	No particular expertise	No particular expertise

Table 2: Definition of Expertise

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

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 as mentioned in the [CEM]. In very specific cases, the "Multiple Expert" level could be used but it should be noted that the expertise must concern fields that are strictly different like for example expert in two or more out of the following domains (non-exhaustive list):

HW manipulation, software attacks, cryptography, fault injection, side channel analysis, reverse engineering.

	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

24 Knowledge of the TOE refers only to classification levels related to the identification and exploitation of vulnerabilities in the TOE.

25 Care should be taken to distinguish information required to identify the vulnerability from the information required to exploit it, especially in the area of sensitive or critical information. 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. To require sensitive or critical information for exploitation would be unusual.

26 The protection of the information will determine the classification of the information.

27 The knowledge of the TOE may graduate according to design abstraction, although this can only be done on a TOE by TOE basis. Some TOE designs may be public source (or heavily based on public source) and therefore even the design representation would be classified as public or at most restricted, while the implementation representation for other TOEs is very closely controlled and is therefore considered to be sensitive or even critical.

28 For the dissemination of information outside the developer organisation a distinction can be made between distributing information and providing access to information. Distributing information means handing over the information, thereby its use can not be (access) controlled anymore by the developer. Providing access means that the information will remain under the developer’s control and its access will be controlled and protected. Different degrees can exist for distribution and access, as defined below.

29 The higher the classification, the more difficult it will be for an attacker to retrieve the information required for an attack. This specifically applies to all sensitive and critical information where a site audit is required to provide the necessary assurance on the sufficiency of security measures (See also [CC] ALC_DVS.2 if applicable).

30 Note that the developer organisation is defined as all organisations that are involved in the development and production phases of the product life-cycle that is subject of the evaluation (See also [CC] ALC class). This means that e.g. a mask manufacturer subcontracted by the smart card developer is considered to be part of the developer organisation and its protection and access control measures are part of the evaluation.

31 Note: Since this document defines the rating of attacks, the sharing of information during the evaluation with a trusted system of Certification Bodies and recognized

ITSEF(s) does not influence the classification below. A trusted system means that all Certification Bodies within this system trust each other.

32 Note: The ETR for composition (ETR_COMP) is a document controlled through the CC scheme which has issued the associated certificate. It is dedicated to be used by an ITSEF evaluating a composite product and does not enter in the rating of the attacks.

33 The following classification is to be used:

- **Public information** about the TOE (or no information): Information is considered public if it can be easily obtained by anyone (e.g., from the Internet) or if it is provided by the developer to any customer without further means.
- **Restricted information** concerning the TOE: Information is considered restricted if it is controlled within the developer organisation and distributed to other organisations under a non-disclosure agreement.
- **Sensitive information** about the TOE is knowledge that is only available to discrete teams[2] within the developer organisation. Sensitive information is protected by evaluated secure IT systems (e.g. through MSSR) and by appropriate environmental and organizational means. If such information needs to be distributed to or accessed by other organisations outside the developer, this must be limited to a strict need-to-know basis protected by a specific contract.
- **Critical information** about the TOE is knowledge that is only available to teams[2] on strict need-to-know basis within the developer organisation. Critical information is physically and environmentally protected by high secure IT infrastructure as well as secure physical environment including attack detection and attack prevention layers. If such information needs to be accessed by other organisations than the developer, this must be limited to a strict need-to-know basis protected by a specific contract.
- **Very critical information** about the TOE is knowledge that is known by only a few individuals[2], access to which is very tightly controlled on a strict need to know basis and individual undertaking. The design of modern ICs involves not only huge databases 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 of the developer organization. If an attack is based on such knowledge the new level of "Very critical information" is introduced.
  Very critical information shall never be shared with organisations outside the developer without consulting the respective Certification Body that issues a certificate.
  It could occur that very critical information cannot be exported for technical reasons as the sophisticated bespoke tools of the developer are required to interpret the information or there is simply no interface for exportation or there is only a dedicated group of people[2] – which can be different to the other groups of lower classification – specifically enabled to access this very critical

[2] All people involved in getting access to such information must be considered in the ALC activities.

information.
A review of such information is therefore usually only possible on the developer's premises. The common understanding of all parties of the evaluation should be that export of such information outside the developer's premises is an exceptional risk that should be avoided.

- Information is considered as *Not practical* if it is maintained by highly secured IT systems only (within sites protected as for very critical and critical information), e.g. some personalization keys handled by HSMs only.

34 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	9	*
Not practical	*	*

Table 5: Rating for Knowledge of TOE

35 Note: this section is currently under discussion and rework and will be updated in a future version of this document. For national interpretations please consult the respective Certification Body.

4.5 Access to TOE

36 Access to the TOE is also an important factor. It is assumed here that the samples of the TOE would be purchased or otherwise obtained by the attacker and that beside other factors there's no time limit in analyzing or modifying the TOE. The 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 "Window of Opportunity").

37 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.

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

38 "Not Practical" is explained as follows:

- For identification: not practical starts with the lowest number between 2,000 samples and 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 the lowest number between 500 samples and the largest integer less than or equal to n/(1+(log n)^3), n being the estimated number of products to be built.

39 As an example, if n equals 20,000 (samples produced), the "Not practical" limits would be 1,025 and 248 samples respectively for identification and exploitation.

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

4.6 Equipment

41 Equipment refers to the hardware/software or cloud/online services that are required to identify or exploit the vulnerability.

42 In order to clarify the equipment category, price and availability have to be taken into account.

- **None**
- **Standard equipment** is equipment that is readily available to the attacker, either for the identification of vulnerability or for an attack. This equipment can be readily obtained e.g., at a nearby store or downloaded from the Internet. The equipment might consist of simple attack scripts, personal computers, card readers, pattern generators, simple optical microscopes, power supplies, or simple mechanical tools.
- **Specialized equipment** is not readily available to the attacker, but could be acquired with increased effort. This could include purchase of moderate amounts of equipment (e.g., power analysis tools, use of hundreds of PCs linked across the Internet, protocol analyzers, oscilloscopes, microprobe workstation, chemical workbench, precise milling machines, etc.) or development of more extensive attack scripts or programs.
- **Bespoke equipment** is not readily available to the public as it might need to be specially produced (e.g., very sophisticated software) or because the equipment is so specialized that its distribution is controlled, possibly even restricted. Alternatively, the equipment may be very expensive (e.g., Focused Ion Beam, Scanning Electron Microscope, and Abrasive Laser Equipment). Depending on the possibilities of renting equipment and the type of manipulation to be performed, the classification of the equipment as bespoke might be reconsidered. Complex and dedicated software (e.g. advanced analysis tools that are not available for purchase) that has been developed during the identification phase can be considered as bespoke equipment or alternatively rated according to Elapsed time and Expertise criteria; it must not additionally be considered in the exploitation phase. If an evaluator has to adapt his dedicated analysis software, e.g. alignment tools/scripts or filters

specifically to the TOE or TOE derivatives, then this has to be rated extra (Elapsed time, Expertise, Knowledge of the TOE, samples …) in the identification phase.

Complex and dedicated software as introduced above can be characterised as being developed during the identification phase for the TOE under evaluation, or applied to another TOE while the ITSEF still considers it as beyond the state-of-the-art. In case there is uncertainty about the state-of-the-art, then discussion might be required at JHAS level.

43 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.

44 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

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

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

47 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 7: Rating for Equipment

(1) If clearly different test benches consisting of specialized equipment are required for distinct steps of an attack this shall be rated as bespoke. Test benches for side-channel and fault attacks are normally considered to be too similar and not different enough. In such cases where multiple similar specialized equipment is required, this will then be considered as Multiple Specialized and an additional 1 point will be added to the rating.

4.6.1 Tools

48 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 breakdown just provides 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 analyser	Specialized
Tools for chemical etching (wet)	Specialized
Tools for chemical etching (plasma)	Specialized
Tools for grinding	Specialized

Table 8: Categorisation of Tools (1)

4.6.2 Design verification and failure analysis tools

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

50 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 9: Categorisation of Tools (2)

4.7 Open Samples/Samples with known Secrets

4.7.1 Purpose of this section

51 In a composite evaluation as a rule, the properties of the underlying platform are taken from the information supplied with the documentation from the certification of the underlying platform. For this purpose, the JIL document [COMPO] specifies the process, called “composite smart card evaluation”. In that document, platform and application are relative notions and generic terms. Platform can be a certified IC with or without Embedded Software that is used as the underlying foundation for the composite evaluation. Application designates an additional piece of Embedded Software that is added on top of the certified platform. It can be for example an Operating System, an application on a Java Card product or a combination of Operating System and Applications. The notions of platform and application that are used in the sections below correspond to those definitions.

52 In general, the “ETR for composite evaluations” (ETR_COMP) 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.

53 Note: this section is currently under discussion and rework and will be updated in a future version of this document. For national interpretations please consult the respective Certification Body.

4.7.2 Definition of “open samples / Samples with known Secrets”

54 Within the context of a composite evaluation, the term “open samples” stands for samples where the evaluator can put applications on top of the platform at his own discretion that bypasses countermeasures prescribed in the platform guidance. The intention is to use test applications without countermeasures but not deactivate any platform 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 application should serve to highlight platform properties described in the ETR_COMP considering the special use of the platform in the TOE but not be used to repeat the platform evaluation. If the platform allows different configurations, the configuration implemented in the TOE shall be used. With these samples, it is thus possible to characterise the platform without applications.

55 “Samples with known secrets” refers to a TOE for which the evaluator knows or can define one or more pieces of secret data, such as a PIN or key for performing either passive (monitoring) or fault attacks, yet without deactivating any countermeasures.

4.7.3 Use of “open samples / Samples with known Secrets”

56 For a composite evaluation, the TOE is the combination of platform and application 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 platform part in a composite evaluation this knowledge is provided by the evaluation results as described in the JIL document [COMPO].

57 The documents passed on from the platform evaluation to the composite evaluator describe the protection against threats and state requirements on the environment (especially the application) necessary to obtain this protection. In addition, these documents will be a guidance on how the platform has to be used to achieve the security objectives.

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

59 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 application's countermeasures.

60 Also, the platform may be used in a way that was not foreseen by the platform developer and the composite evaluator, or the application developer may not have followed the recommendations provided with the platform and implemented different countermeasures where the effectiveness is not yet proven.

61 Finally, the composite evaluator has to consider parts of the platform functionality that may not have been covered by the Security Target of the platform and therefore the previous platform evaluation.

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

- The composite evaluator can consult the evaluator of the underlying platform and draw on his experience gained during the evaluation.
- Separation of vulnerabilities of application and platform with the use of "open samples"and/or the use of "samples with known secrets".

63 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 platform 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 underlying platform evaluation in accordance with the rules (see R48 of [COMPO]).

- The rating should make provision for the judgement whether or not the attack would have been possible without the use of “open samples”.

4.7.4 Implications on the composite evaluation

64 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.

65 A good example for this is the retrieving of secret information (e.g. keys) by light attacks. In a well-designed product, the platform as well as the application 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 application 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.

66 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 HWplatform evaluation documentation). With the knowledge gained in these tests the attacker can then launch much more directed attacks on the TOE.

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

4.7.5 Implications on the composite rating

68 For the rating two possibilities have to be considered:

- Freely programmable samples of the platform 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 platform evaluation. In that case, the rating has to include an additional factor for the use of “open samples” as described in the table below.

4.7.6 Background of the use of “samples with known secrets” to accelerate the evaluation

69 An additional possibility to accelerate the evaluation especially where cryptographic operations are involved is the use of “samples with known secrets”. 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”.

70 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 secrets”.

71 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.

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

4.7.7 Calculating the attack potential

73 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.

74 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).

75 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 secrets”.

76 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 secrets). Having determined this, the evaluator will perform two calculations, using open samples / samples with known secrets:

- 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

77 Should it turn out that the attack is not practical when not using open samples or samples with known secrets, then that rating has to be used and the rating with open samples / samples with known secrets must be 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.

78 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.

79 For platforms, the protection level will be analysed during the underlying platform evaluation and stated in the ETR_COMP.

80 For "samples with known secrets", defining the protection level is part of the evaluation of the full product.

81 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 and decision rules 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 platform is used in combination with 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 user deemed specifications of the platform, as the data sheet of an IC, or delivered without additional 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: Similar protected as the restricted knowledge, i.e. controlled within the developer organisation and distributed to other organisations under a non-disclosure agreement..
  - 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 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.

82 The composite evaluation has also to define if the use of “open samples” **and** “samples with known secrets” 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.

83 The platform evaluation will give a rating for the “open samples” in the ETR_COMP. Any indication for a different rating has to be considered in the composite evaluation.

	Identification	**Exploitation**
Public/Not required	0	NA
Restricted	2	NA
Sensitive	4	NA
Critical	6	NA

Table 10: Rating for Open Samples/Samples with known secrets

4.7.8 Impact on the evaluation of an IC platform with guidance

84 As such, the concept of “samples with known secrets” does not apply to IC platform 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.

85 The situation is more complex with regards to the concept of “open samples”. The evaluation shall consider the following 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 IC platform evaluations, since there is no way of knowing whether a final product based on this IC platform does implement all SW countermeasures recommended in the IC guidance in such a way that no loophole could possibly exist.

4.7.9 Impact on the evaluation of a firmware / crypto library application of an IC platform with guidance

86 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.

87 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.

88 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.

4.8 Calculation of attack potential

89 Table 11 identifies the factors discussed in the previous sections and associates numeric values with the two aspects of identifying and exploiting a vulnerability. It replaces Table B.3 of [CEM] for products that fall under the technical domain of “Smart Cards and similar devices”.

Factors	Identification	Exploitation
Elapsed time		
< one hour	0	0
< one day	1	3

Factors	Identification	Exploitation
< one week	2	4
< one month	3	6
> one month	5	8
> four months[3]	6	10
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	9	*
Not practical	*	*
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 11: Final table for the rating factors

90 (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. In such cases where multiple similar specialized equipment is required, this will then be

[3] See paragraphs §14-15 for applicability of this factor.

considered as Multiple Specialized and an additional 1 point will be added to the rating.

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

92 The following table replaces Table B.4 of [CEM] and should be used to obtain a rating for the vulnerability.

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 12: Rating of vulnerabilites and TOE resistance

93 *final attack potential = identification + exploitation.

5 Examples of attack methods

94 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).

95 The collection represents the current state of the art at the time. As state of the art is not static this document is under review of the same expert group and will be updated if necessary.

96 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.

97 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

5.1.1 General description

98 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.

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

5.1.2 Impact on TOE

100 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).

101 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

102 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

5.2.1 General description

103 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.

104 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.

105 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.

106 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)

5.2.2 Impact on TOE

107 Under this attack, 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).

108 If a chip returns incorrect cryptographic results then this may allow a DFA attack, see section 5.4. Other consequences are described under general perturbation effects in section 5.3.

5.3 Perturbation Attacks

5.3.1 General description

109 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.

110 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.

111 Section 5.3 concerns itself more with the methods to induce meaningful faults whereas section 5.4 describes how these induced faults may be used to extract keys from cryptographic operations.

5.3.2 Impact on TOE

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

113 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 Section 5.7 "Attacks on RNG" 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

114 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).

115 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 FA

5.4.1 General description

116 By using Fault Analysis (FA), an attacker intends to obtain information about a secret key by analysing the difference between a correct and a faulty cryptographic output, , or by analysing different faulty cryptographic outputs.

117 This attack method requires analysing faulty outputs. Such faulty output could be obtained by inducing a physical perturbation on the device during the corresponding cryptographic computation, or eventually during the algorithm parameters manipulation. Such perturbation can be created by either non-invasive (power glitching for instance) or semi invasive (laser typically) techniques.

118 According to the theory behind this attack, the fault injected during the device processing should fulfil specific requirements to lead to an exploitable output. For most attacks, these requirements are based on both a precise synchronisation and the expected value as a consequence of the perturbation. A lack of accuracy in these requirements can render the analysis to recover the key much more complex.

119 From a practical point of view, the process to mount such an attack can then be divided into the following stages:

- Searching for a suitable fault injection method
- Depending on the cryptographic algorithm to attack, setting up a more or less accurate synchronisation technique.
- Inducing fault(s) during the device’s execution and then collecting the corresponding faulty cipher texts
- Analysing the differences between the faulty cipher texts with the correct cipher text (or eventually the plain text).

5.4.2 Impact on TOE

120 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.

121 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.

122 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 analysing the power consumption to determine when the cryptographic computation occurs.

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

5.5.1 General description

123 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.

124 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, usually without damaging the TOE. 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.

125 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).

126 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.

127 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.

128 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.

129 The combined statistical analysis for higher-order DPA may be based on aligned measurements of the same side channel at different times or on aligned simultaneous measurements of different channels such as power consumption and electromagnetic radiation of the device during the computation.

130 The outcome of a side-channel attack may be as simple as finding a characteristic trigger point for launching other attacks (such as DFA), or as complete as the secret key used in a cryptographic operation. It can also aim at recovering other secret data such as PINs, or random numbers generated for use as secrets, or even the opcode of

the code being executed on the TOE. Depending on the goal of the attack it may involve a wide range of methods from direct interpretation of the recorded signal to a complex analysis of the signal with statistical methods. In the latter case the initial filtering used for signal analysis will generally depend on the type of the measurement (i.e., power consumption or electromagnetic radiation), but the mathematics for retrieving the secret information eventually is largely the same.

5.5.2 Impact on TOE

131 It lies in the very nature of SPA and (higher-order) DPA attacks that they may in principle be applied to any cryptographic algorithm – either stand-alone, e.g., for retrieving secret keys or PINs, 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., in the square-and-multiply algorithm of RSA), or for deducing which branch of a conditional jump has been taken by the program flow. Or it could simply be used as a first step for identifying countermeasures to side-channel attacks that need to be overcome. Finally, an SPA attack could be employed to determine the proper trigger point for a subsequent glitch or light attack, or as an aid for localising a suitable time window for a physical probing attack

132 A DPA (or template) 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 as discussed further below, it may be enough to retrieve only part of the secret key by DPA, and obtain the rest by brute-force methods.

133 Implementations that resist DPA attacks may still be vulnerable to higher-order DPA attacks since that type of attack is tapping additional information not considered in a standard attack. Of course, algorithms that are vulnerable to first-order DPA are vulnerable to higher-order DPA, too. It appears that higher-order DPA is particularly suited to deal with Boolean and arithmetic masking / blinding of symmetric algorithms. On the other hand, the extension of higher-order DPA to public key (asymmetric) algorithms seems to be very difficult, because of the widely applied blinding countermeasures that make use of algebraic transformations during the calculation that are completely different from ordinary masking.

134 Power analysis as well as EMA attacks may be carried out for a hardware TOE (an IC), or a software/composite TOE (an OS or application running on an IC). Some countermeasures may already exist in the hardware TOE, whilst others are added later in software. Thus, the way in which software uses the IC functions may make a critical difference to its vulnerability to this type of attack.

5.6 Exploitation of Test features

5.6.1 General description

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

136 These further attacks might lead for example to disclosure or corruption of memory content, a change in the lifecycle state, or deactivation of security features. But as this depends on the possibilities of the test mode, the details about those further attacks are not considered here.

5.6.2 Impact on TOE

137 As result of successful access to the IC test mode , the attacker might be 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.
- 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

5.7.1 General description

138 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) .

139 A RNG considered here can be one of the following types[4]:

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

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

141 A true RNG may be attacked by[5]:

- permanent or transient influence of the operating conditions (e.g. voltage, frequency, temperature, light)

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

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

- 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 tapping internal signals (compromise internal states)

142 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)
- exceed 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)

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

144 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.

5.7.2 Impact on TOE

145 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. if sensors and related tests being disabled by an attacker then this will generate further attack possibilities.

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

147 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 the case of PRNG, special effort on protecting the seed and the algorithm in terms of integrity and confidentiality

is required. This effort relates to general software and data protection aspects and will not be discussed further in this chapter.

5.8 Ill-formed Java Card applications

5.8.1 General description

148 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.

149 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 samples). In addition, if the card includes an embedded byte-code verifier, this verifier must be disabled. No other specific configuration is required.

150 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 do not enforce the rules during the execution of applications.

5.8.2 Impact on TOE

151 In the most successful 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 full control over the card (JCRE context).

5.9 Software Attacks

152 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 or extensive software testing. Both are usually combined for more efficiency on the coverage of vulnerabilities detection.

153 In general, it is important to note that most software attacks arise from:

- errors (bugs) in the TOE, either in design or implementation;
- inconsistency, holes or ambiguity in the specification or standards;
- exploitation of sensitive or critical knowledge obtained on the TOE.

154 In the case of errors (bugs), it will generally result in a failure to meet the requirements of one (or more) of the ADV families. 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).

155 In the case of issues coming from the specification or standards, the modification of the 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.

156 In the case of exploitation of knowledge of the TOE, the attacker may have access to authentication data for example, opening the usage of some features only accessible for the developers. For example, the attacker may use proprietary administration commands requiring authentication.

157 This section therefore lists a number of attack steps that may be used to discover software errors. If any error is discovered then it must be corrected if the TOE is to pass evaluation.

158 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:

- Information gathering on commands
- Direct protocol attacks
- Man-in-the-middle and replay attacks
- Buffer overflow or stack overflow
- Communication interface switching

159 Attacks related to application isolation (loading, firewalls, etc.) are not described in this section but in section 5.10 “Application isolation”

160 The attacks are of a logical nature, 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)
- a means to analyse the source code with a tool
- a means to create applications
- a means to build and run larger test suites

161 So the test environment may consist of :

- A PC
- a smart card reader
- test software for test scripts writing and execution and communicating with the smart card
- a source code analysis tool (for white box testing or memory dump analysis)
- a protocol analyser (for reverse engineering of communication protocols)
- a development environment (for development of applications to load on the TOE)

162 Setting up a test environment and identifying an attack could be done in rather short time, as the following applies:

- the tools are considered to be standard equipment (some software tools are even available as freeware on the Internet),

- the commands are often ISO standard and therefore public knowledge,

However:

- tools usage and interfacing with the equipment for building the test scripts may require some significant set-up time,
- if the command set is proprietary, the expertise needed is slightly higher because the communication must be interpreted.

163 Note that if the security level is based on 'security by obscurity', it would not be considered a valid defence against attack.

164 The expertise of the attacker could be proficient or expert, and may be multiple expert, especially when combining very precise areas of expertise (Java Card and cryptography for example).

5.9.1 General description

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

166 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.

167 Executing commands that are not allowed in the current lifecycle state of the Operating System or of a data object can also be obtained by various techniques:

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

168 The manipulations of lifecycle state information and access rules require a logical attack on the smartcard and its Operating System and applications. The circumvention and manipulation of the request and evaluation of lifecycle state information and access rules is based on a manipulation of the intended program flow that may be achieved by logical means (physical means are not considered in this example).

169 In the rest of this section, different type of software attacks techniques are described and have to be considered as elementary building bricks: usually, a full attack path is a combination of the different techniques.

5.9.2 Information gathering on commands

5.9.2.1 Overview

170 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.

171 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 (exposure of secret data in this way would generally be considered a sufficient vulnerability to cause the TOE to fail evaluation[6]).

172 This attack step results in gathering information on the operation of the TOE. 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.

5.9.2.2 Attack Step Descriptions

Observing Message Sequences

173 Observing message sequences may result in:

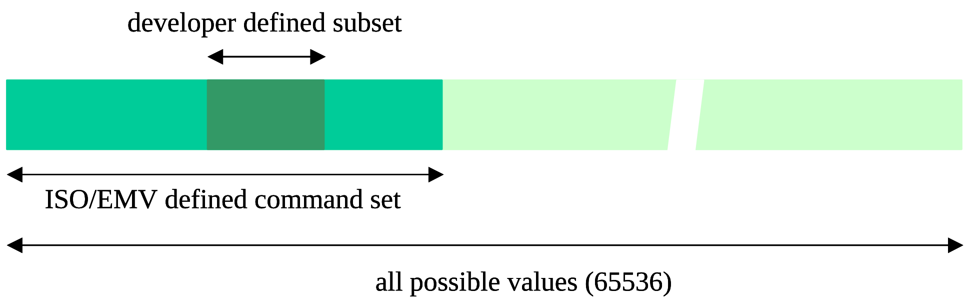
- obtaining information on an unknown protocol (e.g. where the interface specification is not public) to prepare an attack
- obtaining information on unknown internal product structures (typically data structures in software) to prepare an attack
- disclosing information, keys, or security attributes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

174 Such observation is only possible when intercepting a valid communication between a smart card and a terminal. If the attacker does not have such possibility, he has to proceed with the next step "Command searches".

[6] 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.

Command searches

175 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.



176 A T=1 test plan may 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.
- Trying all developer documented commands, but with emphasis on limit cases and multiple error cases.
- Influencing the communication by sending commands in different sequences.
- Interrupting message from system or from product

177 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.

178 However, if source code is not available, simple command search of valid CLA/INS pair is not sufficient, as especially in the context of identification of existing commands: sometimes all CLA/INS/P1/P2/Lc have to be correct. So it represents 2^{40} or 1 099 511 627 776 different possibilities (see ISO/IEC 7816-4 standard).

179 Though an undocumented command search can be highly standardized and automated, the identification could be brief or very costly in terms of time, or even too costly to be considered as practical. Once all variations of parameters are tried and the answers are 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 discover the behaviour of the identified command and exploit a potential vulnerability. This could also be done by source code checking. Note that finding a single command may not be

sufficient, as the attacker may have to look for a specific sequence of commands, sometimes following a proprietary protocol.

180 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.

<u>Editing commands</u>

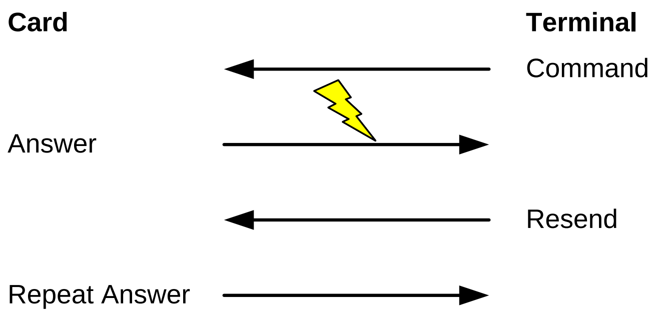
181 Editing commands is an attack step where the attacker tries to modify commands during the communication sequence to see if the card gives an unexpected reply (these commands may be in an interface specification, or they may have been discovered by observing message sequences or a command search as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

182 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.

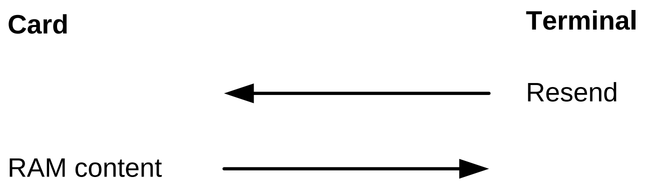
5.9.3 Direct protocol attacks

183 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)



184 In this example, 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 recently calculated DES session key. Therefore this attack may allow an attacker to retrieve secret data from the TOE.

185 Under the category direct protocol attacks, there are also attacks focusing on the state machine of the TOE, where some sensitive operations need a specific order. Such order may ensure that the keys used in the cryptographic calculations are not exposed (such as challenge sending before a signature).

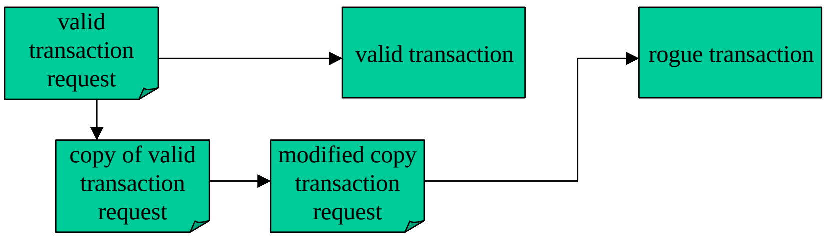
5.9.4 Man-in-the-middle and Replay attacks

186 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

187 Man in the middle attacks are based on valid command interception either to perform replay attacks or to change some of the parameters to compromise the exchange of data (get access to confidential data exchanged, modify the parameters exchanged).

188 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[7]). 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

189 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

[7] 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.

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.

190 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. as a result of the attack the attacker may be able to communicate as an ordinary user and may then try to gain privileged status).

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

5.9.5 Buffer overflow or stack overflow

192 This attack is applicable to any embedded software or firmware. An example is given below for an open platform. Open platforms are defined in this document as smart card operating systems with the capability of running and downloading multiple applications.

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

194 This attack could be performed through buffer overflow or stack overflow, produced by the execution of a malicious application. Overflow, when not checked by the platform, can have various effects, such as overwriting existing content in the current stack.

195 The expected effect by the attacker here is that the malicious application modifies the current execution context and obtains system privileges. For example, the execution rights of the current context is written in the stack; if the attacker can overwrite these rights and put the administrator rights, all operations performed after this operation will have the administrator rights. As another example, the attacker may overwrite a memory location that contains a pointer in memory. The attacker may then control where an application is getting its data.

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

197 Another effect is that internal data or data that were not presumed to be returned by a command are retrieved.

5.9.6 Communication interface switching

198 This attack is applicable to dual interface cards. The purpose is to exploit the possibility to communicate with a TOE on two different interfaces.

199 For example, on a TOE inserted in a mobile phone with a NFC interface, the TOE can be accessed either by the NFC interface or by the applications downloaded in the phone (communication in contact mode). The attacker may wait for a valid mutual authentication between the terminal and the TOE, and then through the contact interface, the attacker could communicate with the TOE to take advantage that a

secure session is opened. This is a way for the attacker to bypass a state machine chaining and to get access to commands with a privileged access.

5.10 Application isolation

5.10.1 Introduction

200 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.

201 The combination of the physical and logical measures of the multi-application platform allows the application isolation, which might be defined as: all the security measures and mechanisms which protect the sensitive assets from the application and the platform against modification and/or disclosure.

202 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.

203 Applet isolation is the target of various types of attack techniques to reach these assets. As such, the multi-application platform is subject to the typical Smart Card attacks, such as:

- Physical attacks,
- Perturbation attacks,
- Side Channel attacks and,
- Software attacks, which may or may not be combined with the above-mentioned attacks, detailed as:
    - Unauthorized disclosure of Loaded Application data (such as application data, code and keys).
    - Unauthorized use of instructions, commands or sequence of commands.
    - Bypassing the Administrator restrictions by loading a malicious application on the multi-application platform.
    - Read confidential data or code belonging to another Loaded-Application without authorization by using a Loaded-Application.
    - Modify data or code belonging to another Loaded-Application without its authorization, by using a Loaded-Application.
    - Access the confidential data of system resources (like a system patch), by using the Loaded-Application.
    - Reverse-engineer the abstraction layer mechanisms by Using the Loaded-Application

5.10.2 Partial attacks

204 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.

205 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.

5.10.3 GlobalPlatform partial attacks

206 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 an additional layer to the JCRE which provides extra management functionalities to the card. If present on the card, the OPEN is responsible for command dispatching, (optional) multiple logical channel management, management of application and card lifecycle.

  The OPEN provides also a concrete management for performing the installation and deletion of applications. For this, the GP specification defines the notion of a security domain.

- Security domain (SD)

  A security domain represents a smart card actor on the card. Three main actors may be present on the card: an application provider (AP), a card issuer (CI) and a controlling authority (CA). SD is a special kind of a smart card application which 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. A SD stores cryptographic secrets of the actor which it represents. A GP card is always provided with a CI security domain. One of the benefits which SDs bring to security providers is that an AP may benefit of a certain independence with respect to the CI mainly for provisioning of security services (e.g. secure messaging, or application personalization) to its associated applications. A SD may be granted further privileges which would enable it to perform Card Content Management (application load, installation etc.).

- Cardholder verification methods

  These are the common security services which the card provides to all applications. In particular, this gives the possibility for a unique user PIN number to be used by all applications.

5.10.3.1 Description of a partial attack example

207 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.

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

209 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. If the implementation is basic, this may consist in a simple verification such as:

```
if (securityLevel == SecureChannel.NO_SECURITY_LEVEL)
      ISOException.throwIt(0x6985);
```

210 Then, the attacker simply needs to send a content management command (without previous `INITIALIZE UPDATE` and `EXTERNAL AUTHENTICATE` command) and if it specifies a 0x80 CLA byte, no secure channel unwrapping will be processed: therefore no cryptographic computations have to be attacked.

211 For a successful applet loading and installation, two distinct sequences are required:

- A sequence for code loading, consisting in an `INSTALL [for load]` and one or several `LOAD` commands;
- A sequence for applet instantiation, consisting in an `INSTALL [for install & make selectable]` command.

212 In the first sequence, the attacker shall be able to reproduce the attack successfully on all the commands within the sequence as the loading operation is atomic (at least 2^{8}). If the attack fails (which generally implies a power off), the sequence shall restart from the initial `INSTALL [for load]` command.

5.10.3.2 Impact on TOE

213 The direct impact is that the TOE may contain malicious code that could disclose or alter other applications.

5.10.4 Bytecode verifier partial attacks

214 All bytecodes must be verified before their execution in order to avoid the execution of malformed applets. In the Java technology, the ByteCode Verifier is used to verify the class file on the Java Virtual Machine and is operating dynamically (ex: applied each time a class is loaded). However, the full ByteCode Verifier is often not implemented in a Java Card due to its limitation in processing power and memory size. There are several solutions to resolve this issue:

- The application can be verified off-card by an Off-Card Verifier which is not limited by Java Card constraints.
- The application can be verified on-card with a specific On-Card Verifier designed for Java Card.

[8] An optimized malicous applet that is able to execute code loaded into the heap (*e.g.*, in arrays content) can fit in a single LOAD APDU command. Otherwise, an average of about 2 or 3 LOAD commands is to be considered.

215 The Java Card Protection Profile specifies that all bytecode should be verified before its execution. Additional verifications to ensure that the application does not contain malicious code are also required. If all verifications succeed, the CAP file can be loaded onto the card.

216 In this context, the TOE is the smart card because all the assets to protect are in it. There is no asset in the Off-Card Verifier. In fact, this attack allows an attacker to ask for loading its malicious applet without being detected.

5.10.4.1 Description of a partial attack example

217 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. The two following attacks are 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.

  Several steps are necessary for this attack. First, it is needed to characterise the On-Card Verifier in order to understand its behaviour and to analyse checks performed by this tool. Secondly, it is needed to write the malicious applet by creating a type confusion not detected by the On-Card Verifier.

  The main assumption of this attack is that the application could be loaded on card. If it is not the case, this attack will become a combined attack. In fact, the evaluator should use an attack described in section 5.10.3 “GlobalPlatform partial attacks” in order to bypass the loading mechanism.

- Using a well-formed CAP file abusing the transaction mechanism in order to create a type confusion.

  The aim of this attack is to create a type confusion using a weakness in the implementation of the platform enabling us to dump and modify a part of the memory content. This type of attack uses a well-formed CAP file and abuses the transaction mechanism in order to create a type confusion.

  The main assumption of this attack is that the application could be loaded on card. If it is not the case, this attack will become a combined attack. In fact, the evaluator should use an attack described in section 5.10.3 “GlobalPlatform partial attacks” in order to bypass the loading mechanism.

5.10.4.2 Impact on TOE

218 The impact of the type confusion attack is dependent of software implementation.

219 The main impacts are:

- Retrieve secret data (such as cryptographic keys).
- Read data/code outside our context.
- Modify data of another application.
- Modify the source code of another application.

5.10.5 Defensive virtual machine partial attacks

220 There are two approaches in maintaining type safety within virtual machines

- Semi-defensive virtual machine: all bytecodes are either verified before or during installation (off-card or on-card)

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

- Defensive virtual machine: type safety is enforced at run-time because the virtual machine only references typed data

  The defensive virtual machine[9] can analyze the bytecode dynamically during the APDU execution (ex: type verification and structural verification) and does not require off- or on-card bytecode analysis to prevent type confusion.

5.10.5.1 Description of a partial attack example

221 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.

222 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.

223 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.

5.10.5.2 Impact on TOE

224 The attack is directed against other applications installed on the TOE, or the operating system. The main impacts are:

- Access to secret data of the target applet,
- Modification of applet functions and status

225 As the internal representation of data is not public the attacker should have critical knowledge of the TOE to interpret the retrieved data, or by experimental analysis on open samples derive the meaning of the data.

5.10.6 Firewall partial attacks

226 The Java Card Operating System is designed to run all applets in a single virtual machine. It does not have resources to provide a per-application virtual machine, which would provide an isolated runtime environment for each applet. The Java Card firewall is introduced to provide the sandbox environment for applets running in the same virtual machine.

227 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

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

Shareable Interface Object mechanism. Static members are excluded from firewall control and their accessibility does not depend on contexts.

5.10.6.1 Description of partial attacks

228 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.

5.10.6.2 Impact on TOE

229 The attack is directed against other applications installed on the TOE, or the operating system. The main impacts are:

- Access to secret data of the target applet,
- Modification of applet functions and status

230 The potential use of these techniques is specialized and has to be considered in the context of each evaluation. As the internal representation of data is not public the attacker should have critical knowledge of the TOE to interpret the retrieved data, or by experimental analysis on open samples derive the meaning of the data.

5.10.7 Multos partial attacks

231 MULTOS platform provides a secure environment for application execution and data storage. It is a multi-application operating system enforcing applications segregation. MULTOS applications can be developed in C language, in MULTOS Assembler (MEL) or in Java.

232 MULTOS does not have a verifier tool for MEL code because this language is less complex. However, MULTOS has similar security mechanisms such as firewall and secure application loading.

233 MULTOS implements the following countermeasures:

- Instructions, primitives and APDU commands do not allow addresses manipulation. In fact we cannot assign a new address to a variable contrary to Java Card (for instance: `aload_1 astore_3`)
- The Firewall: applet isolation, code space and data space isolation (for instance, we can't perform a jump from code to data). That’s why an application cannot access to another application space and so cannot be accessed by other applications.
- The Application loaded on card can contain:
  - o MEL instructions
  - o Data
  - o DIR record: information about the name of the application when loaded on the card
  - o FCI record: information that is returned when a MEL application is selected

  - Application signature (if exist)
  - KTU (if exist)
  - …

  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).

- The MULTOS Application Abstract Machine provides each application with its own memory space. In fact, the memory space is always relative to the current running application. Tagged addresses are used instead of absolute addresses. This tagged address consists of:
  - A register: ST and SB for static memory, DT, DB and LB for dynamic memory, PB and PT for public memory
  - An offset

  A different instruction will be generated depending on register used. For instance:
  - Instruction "`LOAD SB[1], 0x10`" will be "`39 10 00 01`".
  - Instruction "`LOAD PB[1], 0x10`" will be "`3E 10 00 01`".
- The Loaded application can be encrypted

5.10.7.1 Description of partial attack

234 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.

235 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.

5.10.7.2 Impact on TOE

236 The main impacts of this attack are:

- Retrieve secret data (such as cryptographic keys),
- Read data/code outside our context.

5.10.8 Full attack path

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

5.10.9 Attacks on memory management (getting a resource from another application)

238 This attack is the combination of:

1. Getting a memory dump to locate assets and/or sensitive code through physical attacks or software attacks

The attacker is able through physical perturbation during bytes emission to force the TOE outputting more bytes than expected. The memory dumps obtained, for instance during the ATR or in public APDU commands returning a significant number of bytes, may allow the attacker to identify assets of other applications and their respective addresses in memory.
It shall be noticed that software attacks such as those described in “Bytecode verifier partial attacks” or “Defensive virtual machine partial attacks” could be used to perform such memory dump instead.

2. Loading an applet through “GlobalPlatform partial attacks”.
   The attacker is able through this attack to load a malicious application onto the TOE.
3. Type confusion to manipulate the objects identified in step 2 through “Bytecode verifier partial attacks” or “Defensive virtual machine partial attacks”.
   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 to execute the getKey method on the object through “Firewall partial attacks”.
   The attacker uses physical perturbations to bypass Java Card/Multos Firewall restrictions while manipulating objects out of the legitimate bounds. On a Java Card platform, the malicious applet may illegally invoke the `getKey` method on an address of an object of another context.

239 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 sections, 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.

5.10.9.1 Impact on TOE

240 Any platform security mechanisms could be bypassed to disclose or alter secrets since security routines (decrypt, update, *etc*) are forced to be legally exercised in a context belonging to the attacked application.

5.10.10 Attacks on code execution (calling a code from another application)

241 This section describes an attack similar to the previous one but here applied on a non-shared method of another applet.

242 The same combination of attacks is required, with the following modifications:

1. Getting a memory dump to locate assets and/or sensitive code through physical attacks or software attacks
   Compared to the previous attack, the attacker should not only locate objects (and their respective references) but also reverse part of the code to identify private routines to be called (for instance to reset a security counter or to disable a security mechanism).

2. Loading an applet through “GlobalPlatform partial attacks”.
   Same as previous attack.
3. Type confusion to manipulate the objects identified in step 2 through “Bytecode verifier partial attacks” or “Defensive virtual machine partial attacks”.
   Same as previous attack.
4. Attack on the firewall to execute the getKey method on the object through “Firewall partial attacks”.
   The attacker uses physical perturbations to bypass Java Card/Multos Firewall restrictions while manipulating objects out of the legitimate bounds. On a Java Card platform, the malicious applet may illegally invoke an arbitrary method on an address of an object of another context. However, several perturbations may be required to allow invoking a method on an object not owned by the current context through firewall restrictions as during a method execution, object not owned by the current context may be accessed several times, with each time a firewall check[10].

243 Step 1 and step 2 are used to calibrate the attack. Step 3 and 4 are not recalled here because they are similar compared to the previous attack. It shall be noticed that performing several perturbations is difficult, however still feasible as the firewall check operation can be identified through a synchronisation on the bytecode execution.

5.10.10.1 Impact on TOE

244 A malicious application may access private routines allowing to reset counters or to deactivate security mechanisms.

[10] In 5.10.9, since getKey is implemented at platform level, there is a great chance that the code is in native language and therefore only a single firewall check should be performed.

6 Remaining strength of cryptographic implementations

245 The content of this chapter is currently under discussion by the JIWG and will be provided to JHAS for comments at a later date.

References

[AIS31]	Functionality classes and evaluation methodology for physical random number generator, Version 3, 15.05.2013 (BSI).
[CC]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, April 2017.
[CEM]	Common Methodology for Information Technology Security Evaluation (CEM), Version 3.1, Revision 5, April 2017.
[COMPO]	Composite product evaluation for Smart Cards and similar devices, Version 1.5.1, May 2018, Joint Interpretation Library.