EUROPEAN UNION AGENCY
FOR CYBERSECURITY

EUCC SCHEME

STATE-OF-THE-ART DOCUMENT

APPLICATION OF ATTACK POTENTIAL TO SMARTCARDS AND SIMILAR DEVICES

Version 1.2, August 2023

DOCUMENT HISTORY

Date	Version	Modification	Author's comments
October 2022	1.0	Creation	
June 2023	1.1	Modification to match latest version of the JIL document	Version submitted to the ECCG Based on JIL- Application of Attack Potential to Smartcards and Similar Devices v3.2 (sogis.eu) (changes vs v1.0 highlighted in yellow)
August 2023	1.2	Addition of a document history section with a link to the SOG-IS document the state-of-the-art document is based on, where applicable Clarification of the role of the authorisation of CBs and ITSEFs towards the protection of sensitive information	Based on ECCG comments received Approved for publication on 20/10/2023 by the ECCG and the European Commission

CONTENTS

1 INTRODUCTION 4

2 SCOPE 4

3 FOREWORD: WORKLOAD FOR AVA_VAN.5 EVALUATION 4

4 IDENTIFICATION OF FACTORS 4

4.1 HOW TO COMPUTE AN ATTACK 4

4.2 ELAPSED TIME 5

4.3 EXPERTISE 6

4.4 KNOWLEDGE OF THE TOE 8

4.5 ACCESS TO THE TOE 10

4.5.1 Rating the effort for TOE package preparation 10

4.6 EQUIPMENT 11

4.6.1 Tools 12

4.7 OPEN SAMPLES/SAMPLES WITH KNOWN SECRETS 14

4.7.1 Clarification of the notions of platform, application and HW-TOE 14
4.7.2 Definition of “open samples / Samples with known Secrets” 14
4.7.3 Use of “open samples / Samples with known Secrets” 15
4.7.4 Implications on evaluations 16
4.7.5 Calculating the attack potential 16
4.7.6 Good usage of open samples and guidance for correct rating 19

4.8 CALCULATION OF ATTACK POTENTIAL 20

5 EXAMPLES OF ATTACK METHODS 23

5.1 PHYSICAL ATTACKS 23

5.1.1 General description 23
5.1.2 Impact on TOE 23

5.2 OVERCOMING SENSORS AND FILTERS 23

5.2.1 General description 23
5.2.2 Impact on TOE 24

5.3 PERTURBATION ATTACKS 24

5.3.1 General description 24
5.3.2 Impact on TOE 24

5.4 RETRIEVING KEYS WITH FA 25

5.4.1 General description 25
5.4.2 Impact on TOE 25

5.5 SIDE-CHANNEL ATTACKS – NON-INVASIVE RETRIEVING OF SECRET DATA 26

5.5.1 General description 26
5.5.2 Impact on TOE 27

5.6 EXPLOITATION OF TEST FEATURES 27

5.6.1 General description 27
5.6.2 Impact on TOE 28

5.7 ATTACKS ON RNG 28

5.7.1 General description 28
5.7.2 Impact on TOE 29

5.8 SOFTWARE ATTACKS 29

5.8.1 General description 30
5.8.2 Information gathering on commands 31
5.8.2.1 Overview 31
5.8.2.2 Attack Step Descriptions 31
5.8.3 Direct protocol attacks 32
5.8.4 Man-in-the-middle and Replay attacks 33
5.8.5 Buffer overflow or stack overflow 34
5.8.6 Communication interface switching 34

5.9 APPLICATION ISOLATION 34

5.9.1 Introduction 34
5.9.2 Partial attacks 35
5.9.3 GlobalPlatform partial attacks 35
5.9.3.1 Description of a partial attacks 35
5.9.3.2 Impact on TOE 36
5.9.4 Bytecode verifier partial attacks 36
5.9.4.1 Description of a partial attack example 36
5.9.4.2 Impact on TOE 37
5.9.5 Defensive virtual machine partial attacks 37
5.9.5.1 Description of a partial attack example 37
5.9.5.2 Impact on TOE 37
5.9.6 Firewall partial attacks 37
5.9.6.1 Description of partial attacks 38
5.9.6.2 Impact on TOE 38
5.9.7 Multos partial attacks 38
5.9.7.1 Description of partial attack 39
5.9.7.2 Impact on TOE 39
5.9.8 Full attack path 39
5.9.9 Attacks on memory management (getting a resource from another application) 39
5.9.9.1 Impact on TOE 39
5.9.10 Attacks on code execution (calling a code from another application) 39
5.9.10.1 Impact on TOE 40

APPENDIX A 41

A.1 ACCESS TO TOE FACTOR WITH RESPECT TO PACKAGE REMOVAL 41

A.2 EFFORT LEVELS FOR TOE PACKAGE PREPARATION EFFORT 41

A.3 EXAMPLES FOR RATING THE REMOVAL OF PACKAGES 42

1 INTRODUCTION

This state-of-the-art document supporting the EUCC scheme interprets the Common Criteria Methodology (CEM), 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) of the SOG-IS[1].

It provides guidance metrics to calculate the attack potential required by an attacker to effect an attack on a smartcard or similar device. 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

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

No rigid rules can be given on how much time should be spent on a typical smartcard or similar device AVA_VAN.5 evaluation by a competent ITSEF, but the following guidance shall nonetheless be provided in an effort to harmonise evaluations and the certificates 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 AVA_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.

4 IDENTIFICATION OF FACTORS

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

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.

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 ITSEF 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.

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

[1] https://www.sogis.eu/uk/detail_operation_en.html

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 under the EUCC scheme. 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.

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[2]. This means that the elapsed time, expertise and TOE knowledge ratings for exploitation will sometimes be lower for exploitation than for identification. For example, it is assumed that the script identifies such things as the timing 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.

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.

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 Certification Body, with the Certification Body making the final decision if agreement cannot be reached.

4.2 Elapsed Time

Compared to the “Elapsed Time” factor as given in the 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:

Table 1: Rating for Elapsed Time

	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)	*	*

[2] 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.

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:

Table 1a: Extra rating step for Elapsed Time

	Identification	Exploitation
> four months	6	10

It is not reasonable to expect an ITSEF 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.

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

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.

Extrapolation of results is intended to save cost and time, however if the rationale is solid but the developer challenges the results, extra testing needs to be performed. The Certification Body must be informed of such extra testing. Note that it shall be accounted separately from the workload initially scheduled and not replace other attacks.

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.

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.

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

For the purpose of smartcards and similar devices, expertise levels are defined based on the attacker’s ability to implement attacks, devise attack paths, develop attack setups and procedures, as well as the capability to understand the attack concepts, when applied to at least one out of the following domains (non-exhaustive list): HW manipulation, software attacks, cryptography, fault injection, side channel analysis, reverse engineering. Another factor defining the expertise level is attacker’s capability to operate necessary tools and equipment (for a list of tool and equipment examples please refer to Table 9).

Table 2 contains detailed definitions and differentiating factors for the Expertise levels. In particular, the Expert attacker has the ability to not only understand complex concepts, but also to use this understanding in order to innovate and adapt. This includes creating new attack techniques, new attack paths, non-off-the-shelf setups or procedures, as well as, redesigning or adapting existing complex attack techniques, attack paths or procedures to apply them to the TOE. Innovation and adaptation capabilities are not expected from the Proficient attacker. The Proficient attacker’s capabilities are limited to parameter adjustments such as the ones described in user manuals for benches and tools.

Table 2: Definition of Expertise

	Definition according to CEM	Detailed definition to be used in smartcard or similar device evaluations[3]
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.	• Having a capability to implement newly published attacks (typically based on a paper or a related patent), or a capability to devise new attack techniques or attack paths not addressable by off-the-shelf benches and tools and well prescribed and available sets of procedures. This also includes redesigning or implementing of attack techniques, attack paths, setups or procedures for well-established complex attacks, where the novelty or the need for adaptation is, for example, related to a specific target or implemented countermeasures. and • Having a deep knowledge or extensive training or experience in implemented algorithms, protocols, hardware structures, security behaviour, principles and concepts of security employed that allows for understanding the concepts of state-of-the art attacks and attack procedures. OR Having a capability to operate complex tools and equipment, that require expertise beyond what can be easily acquired from user's manual. This might, for example, include expertise in material science or advanced imaging that is needed for intermediate result interpretation.
b) Proficient	Familiar with security behaviour of the product type	• Having a capability to perform attacks following previously developed and available procedures, where possible parameter adjustments have to be described in detail, such as the ones described in user manuals for benches and tools. and • Having basic knowledge, training, or experience in implemented algorithms, protocols, hardware structures, security behaviour, principles and concepts of security employed. OR Having enough practice and knowledge to operate off-the-shelf benches and tools, relying on available associated user's manuals.
c) Laymen	No particular expertise	No particular expertise

In case of ambiguities, the decisions about Expertise levelling should be made by ITSEF on a case by case basis. In particular, in certain cases such as for HW manipulation, if a set of procedures to perform the attack is well prescribed and available, but is very complex to execute, Expert attacker level can be considered. Conversely, if an exact set of non-complex procedures to perform a sophisticated attack is not prescribed and available, but only insignificantly differs from such available set, the attacker level can be considered as Proficient.

In addition, ITSEF should distinguish between the internal availability of developed attack procedures and their availability outside the ITSEF. This is important, especially in the case of consecutive evaluations of similar products. In particular, the rating should always reflect the difficulty of the entire attack path as if performed by non-ITSEF entities.

Both Proficient and Expert levels can be reached based purely on the ability to operate tools and equipment. An example of tools and equipment resulting in Expert rating are advanced Failure Analysis tools such as e.g. Focus Ion Beam station, which require extensive expertise to operate even when applied to attacks that are well established, well described and non-complex.

[3] The logical operators in this column should be interpreted as: (clause 1 and clause 2) OR clause 3.

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. For example experts, as defined in Table 2, in two or more out of the following domains (non-exhaustive list): HW manipulation, software attacks, cryptography, fault injection, side channel analysis, reverse engineering.

Table 3: Rating for Expertise

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

4.4 Knowledge of the TOE

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

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.

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

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.

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.

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 ((for state-of-the-art protection, see also CC ALC_DVS.2 if applicable and the state-of-the-art document MINIMUM SITE SECURITY REQUIREMENTS).

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 the 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.

Note: Since this document defines the rating of attacks, the sharing of information during the evaluation with authorised Certification Bodies and ITSEF(s) does not influence the classification below, as their authorisation will asses they have the appropriate technical and operational measures and competences in place to effectively protect confidential and sensitive data for assurance level ’high’.

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.

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[4] within the developer organisation. Sensitive information is protected by evaluated secure IT systems (e.g. through the requirements associated with the state-of-the-art document MINIMUM SITE SECURITY REQUIREMENTS) 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[5] 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.
- **Critical+ information** about the TOE is knowledge that is known by only a few individuals[6] 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.
  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 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[7] – which can be different to the other groups of lower classification – specifically enabled to access this very critical 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 Critical and Critical+ information).

Similar to chapter 4.3 table 2 a matching table to the rankings defined in CEM is given in table 4 as there was the need to refine granularity in evaluations for Smartcards or Similar devices and to adjust the definitions from the CEM. The given rankings in a row share the same protection level.

Table 4: Correspondence table of classification levels for Knowledge of TOE

Classification used in CEM	Classification used for Smartcards or Similar devices
Public	**Public**
Restricted	**Restricted**
Sensitive	**Sensitive**
Critical	**Critical** **Critical+**
-	**Not practical**

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.

Table 5: Rating for Knowledge of TOE

[4] All people involved in getting access to such information must be considered in the ALC activities.
[5] All people involved in getting access to such information must be considered in the ALC activities.
[6] All people involved in getting access to such information must be considered in the ALC activities.
[7] All people involved in getting access to such information must be considered in the ALC activities.

	Identification	Exploitation
Public	0	0
Restricted	2	2
Sensitive	4	3
Critical	6	5
Critical+	9	*
Not practical	*	*

4.5 Access to the TOE

Access to the TOE is also an important factor. Generally, it rates the difficulty and effort to access and obtain samples of the TOE and is described in the following. In some cases, the TOE's package may create an additional barrier to access sensitive parts of the TOE. Therefore, the rating of 'Access to TOE' may be extended by considering the package as part of the TOE. The according methodology is described in Section 4.5.1.

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").

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:

Table 6: Rating for Access to TOE

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

"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.

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.

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

4.5.1 Rating the effort for TOE package preparation

For the cases where the vendor defined the package as part of the TOE, the package may consequently be part of an attack path and has to be considered for the identification and exploitation phase.

The following provides guidelines rather than absolute fixed values for the rating, as there is on one hand an uncounted variety of package types and materials and, on the other hand emerging methods and techniques which may not yet be publicly known for removal of those in the field.

Packages may occur, where a removal is difficult regarding methods and techniques and in such cases, the vendor has to support the evaluator with the required information. If there is still uncertainty, the evaluator should get in contact with external specialists, for example universities, institutes etc. in order to get a clear picture of how to rate the removal of such package.

The rating of attacks (e.g. fault injection, reverse engineering, side channel attacks, etc.) is rated independently from the package preparation effort. If the package is claimed to be part of the TOE a partial attack to prepare the package is rated by extra points for 'Access to TOE' as described in the following. This rating of the package preparation effort covers all other factors and therefore no further points shall be given elsewhere.

The guideline for the rating of the package removal considers the deviation into Low, Medium, and High preparation effort of the package. The definition of these terms and rating examples are given in Section A.1.

Table 7: Rating for TOE package removal (extra points in the factor 'Access to TOE')

	Identification	Exploitation
Low preparation effort	0	0
Medium preparation effort	1	2
High preparation effort	2	4

Low preparation effort: Simple packages that can be removed by standard chemical etching, mechanical action, re-wiring, or similar for the attack path.

Medium preparation effort: Packages that have a relatively high risk for fatal damage of the TOE (losing the functionality that is target or required for the evaluation) because of special constructions.

High preparation effort: Packages that require multiple experts, high effort and rare bespoke tooling which are not claimed as security functionality.

Note that if the reverse-engineering does not need to be redone in exploitation, consequently points in exploitation shall only be given if the remaining attack path still requires specialized equipment or above.

4.6 Equipment

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

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 the relevant ECCG level.

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.

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

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.

Table 8: Rating for Equipment

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

[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.

In an ideal world, definitions need to be given in order to know what are the rules and characteristics for attributing a category to an equipment or a set of equipment. In particular, the price, the availability of the equipment (publicly available, sales controlled by manufacturer with potentially several levels of control, may be hired) and the availability of operational resources involved shall be taken into account. Especially, the availability of operational resources involved has to be considered if bespoke equipment such as design verification or failure analysis tools has to be classified.

Manufacturers usually have information about the sophisticated tools market and where such equipment can be procured. Generally, manufacturers control the majority of the second hand tools market as well.

Efficient use of these tools requires either a very long experience or the human operator is hired together with the equipment. In other words, only a small number of dedicated, experienced experts operate the bespoke equipment. 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. Thus some consistency is expected between the ratings for expertise and equipment.

Please note, that in the case that additional operational resources are necessary this has also to be considered within the Expertise factor of the attack ranking table. The tables presented in the next section have been put together by a group of industry experts and will need to be revised from time to time as 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

4.6.1 Tools

The border between standard, specialized and bespoke cannot be clearly defined in all cases. As stated above, this decision shall be made on case by case basis depending on technology state-of-the-art, accessibility of tools, costs for purchasing and operational resources involved.

As a guide for evaluation, the equipment purchase price (whether it is new or refurbished) should be used as main distinguisher according to Table 9. The cost mentioned in this table is not the cost of an attack but only the purchasing market price of each equipment or workstation.

This distinguisher gives the best practical approach considering the equipment availability (respectively procurement). Additionally, it gives each category an assignment. This table will be regularly subject to updates following the equipment market evolution.

Table 9: Equipment rating versus purchasing cost

Purchasing cost	Equipment rating
Up to 10 K€	Standard
Between 10 K€ and 200 K€	Specialized
Over 200 K€	Bespoke

The following Table 10 provides typical examples using the chosen information of Table 9 and the general rules of the previous section and implements a general guideline.

Table 10: Categorisation of Tools

Tool	Equipment
Low-end light injection (UV, flash light)	Standard
Electrical glitches workstation	Standard
Binocular microscope	Standard
Thermal stress tools	Standard
Voltage supply	Standard
PC or workstation	Standard
Software tools (fuzzing, test suite)	Standard
Code static analysis tools	Standard
Low-end oscilloscope	Standard
High-end GPU card	Standard
Signal analysis tools	Standard
EMFI, FBBI workstations	Specialized
Optical microscope	Specialized
3D X-Rays workstation	Specialized
Micro-probing workstation	Specialized
High-end laser workstation	Specialized
Real time pattern recognition system	Specialized
High-end oscilloscope	Specialized
Spectrum analyser	Specialized
Wet chemistry tooling (acids & solvents)	Specialized
Dry chemistry (Plasma)	Specialized
Micro-milling and thinning machine	Specialized
Low-end Scanning Electron microscope (SEM)	Specialized

EM signal acquisition workstation	Specialized
Low-end Emission Microscope (EMMI)	Specialized
Low-end Focus Ion Beam (FIB)	Specialized
High-end Scanning Electron Microscope (SEM)	Bespoke
Atomic Force Microscope (AFM)	Bespoke
High-end Focused Ion Beam (FIB)	Bespoke
New Tech Design Verification and Failure Analysis Tools	Bespoke
High-end Emission Microscope (EMMI)	Bespoke
Chip reverse engineering workstation	Bespoke

4.7 Open Samples/Samples with known Secrets

In certain cases, it is opportune to use special samples within the evaluation process. In the following these samples will be called “open samples” or “samples with known secrets”. The use of the “open samples” or “samples with known secrets”, its scope, and the implications on the evaluation and the attack rating are described in this section.

4.7.1 Clarification of the notions of platform, application and HW-TOE

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. The state-of-the-art document COMPOSITE PRODUCT EVALUATION FOR SMART CARDS AND SIMILAR DEVICES specifies the process, called “composite smart card evaluation”. In that document, platform and application are relative notions and generic terms. A platform can be, for example, a certified IC with its firmware, a certified Embedded Software, or a combination of both. The certified platform is used as the basis for the composite evaluation. An application is the additional Embedded Software that is added on top of the certified platform. It can be, for example, an Operating System on a certified IC, an application on certified IC and Operating System such as an application on a Java Card product or a combination of Operating System and applications on certified IC. The notions of platform and application that are used in the sections below correspond to those used in the state-of-the-art document COMPOSITE PRODUCT EVALUATION FOR SMART CARDS AND SIMILAR DEVICES.

In many cases, the fundament for all subsequent certifications of smart cards and similar devices is the hardware IC certificate. This hardware IC certificate includes at least the HW-IC and firmware to operate it but can include also additional software providing for example cryptographic services to the user. The combination of IC hardware, firmware and additional software comes with dedicated user guidance documents also belonging to the corresponding TOE definition.

Additional software components in a HW IC evaluation use the notion of “application” on top of the HW IC and Firmware. The definition of open samples in the sense of the composite evaluation of applications given in the next chapter, also applies to additional SW components, that are evaluated in the context of a HW IC certification.

For the combination of the HW IC and the firmware, HW-TOE is used as a shortcut in the following sections.

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

Within the context of a HW-TOE evaluation, excluding SW components, the term “open sample” stands for samples with the capability to download and/or run any kind of test software. In addition such samples may allow insecure configurations of the HW-TOE, e.g. to bypass countermeasures of the firmware or to change the internal configuration of the IC hardware. This might include support of specific testing enviroments by the vendor as an operating system package is not included in the HW-TOE. The IC hardware shall not be changed as it would raise validity questions and it is as well not justifiable in terms of cost that the vendor changes the IC hardware just for the purpose of the evaluation.

Within the context of a composite evaluation, or for SW components in the HW-IC certification, the term “open samples” stands for samples where the evaluator can put applications on the platform or the HW-TOE at his own discretion that bypasses countermeasures prescribed in the platform guidance or countermeasures implemented in the applications themselves. The intention is to use test applications without countermeasures but not to deactivate any countermeasures inherent to the platform, or the HW-TOE respectively.

For a composite evaluation, the test application may serve to highlight platform properties described in the ETR_COMP considering the special use of the platform in the TOE but may not be used to repeat the platform evaluation.

In addition, another possibility is to enable the evaluator to define one or more pieces of secret data for an asset of the TOE, such as a PIN or key, where this ability would not be available under the normal operation of the TOE. These samples will be named as “Samples with known secrets” and can be used to perform attacks on this asset without deactivating countermeasures. To enable the evaluator to define secret data for one asset does not mean that this information shall be used to attack another asset of the TOE.

If the normal TOE configuration gives the opportunity to the ITSEF to have full control of input and output data, the use of the term “samples with known secrets” cannot be applied. However, “samples with known secrets” can be considered even during HW evaluation if the vendor gives specific access to internal secrets. For example: cryptographic mechanisms used internally by the HW-TOE, such as used for memory encryption.

Please note, that every functional interface or key necessary for the functional tests of the TOE provided by the vendor to the ITSEF shall not be considered as an “open sample / sample with known secrets”.

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

In some special cases the vulnerability analysis and definition of attacks might result in an attack path that is difficult or in the worst case impossible to be evaluated because it would need considerable time or would require extensive pre-testing, if only knowledge of the TOE is considered.

Additionally, the platform may be used in a way that was not foreseen by the platform developer and the platform 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.

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.

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 with the consent of the platform vendor.
- Separation of countermeasures within the application and countermeasures of application and platform with the use of “open samples”.
- Accelerate the evaluation especially where cryptographic operations are involved by using “samples with known secrets”. With these samples the evaluator knows the “secret” (key). This allows either comparison of retrieved data (e.g. as deduced from passive analysis) against the “known secret”. “Open samples” may be useful in a profiling step required for some attacks such as template 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”.

In order to keep an efficient and meaningful evaluation in a maintainable time as mentioned before, it can be necessary to use “open samples / samples with known secrets”. In such case, certain rules should be followed:

- The purpose of open samples / samples with known secrets is to set up tests for the evaluation and not, in the case of a composite evaluation, to repeat the platform evaluation.
- The use of open samples / samples with known secrets, the information flow between parties and if necessary the support of extra services is discussed and agreed upon between the Certification Body, the evaluator, the developer and the developer of the open samples. This also includes the time spent for tests with the open samples / samples with known secrets.
- Failures and observations resulting from the tests are communicated and made known at least to the Certification Body of the TOE. In case of a composite evaluation, the Certification Body of the composite TOE shall take appropriate steps together with the Certification Body of the underlying platform evaluation in accordance with rules of the state-of-the-art document COMPOSITE PRODUCT EVALUATION FOR SMART CARDS AND SIMILAR DEVICES.
- The rating shall make provision for the judgement whether or not the attack would have been possible without the use of “open samples / samples with known secrets” (see section 4.7.5).

4.7.4 Implications on evaluations

With the use of “open samples / samples with known secrets”, it is possible to enable or 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.

Open samples may allow to perform a leakage assessment prior to any side-channel attack by evaluating the leakage with and without additional countermeasures. Thereafter, the TOE can be validated with an appropriate attack method.

If leakage was found by switching off additional countermeasures and if a theoretical assessment could be done, the number of traces necessary to successfully attack the TOE can be estimated. To get comparable results in evaluations where no leakage assessment is done, the time frame or number of traces of the acquisition campaign has to be limited beforehand and the theoretical estimation has to be compared against this limit.

Another good example for open samples 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 energy, 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.

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. He can also optimise his efficiency at a found “weak spot” before switching back to attack the TOE. Even if one would know about the existence of “weak spots”, still the optimization and the choice of the best spot has then to be done on the final TOE. With the use of “open samples” in these tests the attacker can then launch much more directed attacks on the TOE.

The following examples describe the usage of “samples with known secrets”, for instance:

- To extract the complete key might prove to be very time consuming. With some errors in the retrieved key and no possibility to decide which part of the secret is incorrect, an attack might not be possible due to timing constraints.
- 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 whereas the attack would have been not practical without the use of such samples.

4.7.5 Calculating the attack potential

An additional factor is defined in the attack potential table for “open samples / samples with known secrets” with points given in the identification phase only. Due to the definition of “open samples / samples with known secrets” it is clear that these are forbidden to be used in the exploitation phase.

When rating an attack that makes use of “open samples / samples with known secrets”, the evaluator must first fairly 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 with and without 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 / samples with known secrets” used.

Should it turn out that:

1) the attack is “Not practical” when not using open samples or samples with known secrets, and

2) the rating of the “open sample/sample with known secrets” factor in the field is not public, and

3) the developer formally asserts that the function used on the open sample is not an accessible feature available for users on a device on the field,

then the rating is “Not practical” (i.e. the “open samples / samples with known secrets” rating must be discarded). If the developer changes his formal assessment, a reassessment of the TOE has to be done.

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.

Where “open samples / samples with known secrets” 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 information as defined in the chapter about knowledge of TOE.

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

The points corresponding to the availability of “samples with known secrets” are defined by taking into account the level of access control to the secret provided by the sample and the protection of the secret inside and outside the developer’s organization during the entire life cycle:

- Public:

  The secret is accessible without any restrictions (public documents, sample allowing to know the secret,…).

- Restricted:

  The secret is controlled within the developer’s organization. Outside the developer’s organization all people who have signed the NDA could have access to the secret.

  If the secret can be released by the sample it must be protected by access control with credentials, which are protected as restricted secrets.

- Sensitive:

  Inside or outside the developer’s organization, secrets are only shared by discrete teams or devices clearly identified, with strong access controls. Handling of the secret is governed by specific and appropriate written procedures to protect it, and there is a clear method by which the secret is identified as requiring these procedures (e.g. by labelling the data).

  If the secret has to be distributed to other organisations, this must be on a strict need-to-know basis protected by a specific contract. The other organisation must provide a secure environment which is evaluated or compliant by contract with criteria acceptable by the Certification Body.

  If the secret can be released by the sample, it must be protected by access control with credentials, which are protected as sensitive secrets.

- Critical:

  The Secret is not shared outside the developer’s organization.

  Inside the developer’s organization, secrets are only shared by few people or few devices clearly identified, with strong access controls on a need-to-know basis. Handling of the secret is governed by specific and appropriate written procedures to protect it, and there is a clear method by which the secret is identified as requiring these procedures (e.g. by labelling the data). It could be applied to the following examples:

  - HW Key split between mask and Flash or PUF or other.
  - Signing keys for firmware update in the field.

  If the secret can be released by the sample it must be protected by access control with credentials, which are protected as ‘Critical’ secrets.

- Not practical:

  The secret is not shared outside the developer’s organization.

  The developer has no possibility to know the secret. The sample can not release the secret. For example:

  - Keys completely generated inside the device.

- Keys generated inside an HSM, not accessible by the developer, and transferred to the TOE through secure channel in a secure environment

Table 11: Rating for Samples with known secrets

	Identification	Exploitation
Public/Not required	0	NA
Restricted	2	NA
Sensitive	5	NA
Critical	9	NA
Not practical	*	NA

The points corresponding to the availability of "open samples" are defined by taking into account the number, the protection and control of these open samples during the entire life cycle:

- Public:

  Open samples: No protection of the samples, delivered without control (no NDA, no checking of the customer).

- Restricted:

  Open samples are protected and controlled within the developer's organisation and can be distributed to other organisations under an NDA.

- Sensitive:

  Open samples must be limited in number, protected and controlled within the developer's organisation. If the samples have to be distributed to other organisations, this must also be limited in number and to a strict need-to-have basis protected by a specific contract. The other organisation must provide a secure environment which is evaluated or compliant by contract with criteria acceptable by the Certification Body.

- Critical:

  Critical open samples are never to be distributed outside the developer's organisation. Within the developer's organisation they must be limited in number and are only available to teams on a strict need-to-have basis. Critical open samples are physically and environmentally protected by a secure evaluated physical environment.

The usage of an "open sample" is more powerful than having access to a "sample with known secrets" as it might allow to get access to secrets that are ranked 'Not practical' for the TOE.

Table 11: Rating for Open Samples

	Identification	Exploitation
Public/Not required	0	NA
Restricted	2	NA
Sensitive	5	NA
Critical	9	NA

Please note that sharing of "open samples / samples with known secrets" for the evaluation purpose with a trusted system of Certification Bodies and ITSEF(s) does not influence the classification above.

In specific cases where the "open samples / samples with known secrets" categorization matches an intermediate classification level, the final rating granted for such samples would need to be addressed with the concerned Certification Body(ies) on a case by case basis.

The ITSEF has to define if the use of “open samples” and “samples with known secrets” accumulates the efforts in time during the evaluation and add points for each of them.

For platforms, the protection level of “open samples / samples with known secrets” will be analysed during the underlying platform evaluation and stated in the ETR_COMP.

The wording “Sibling Product” refers to products available in the field that have interesting features in common with the TOE, with less countermeasures activated and/or more functions available. Those products may not have implemented as many countermeasures as the TOE or may have more functions because their security problem is different from the TOE's. When the ITSEF uses a feature from an Open Sample delivered for the evaluation, the Developer provides an analysis addressing the threat of “Sibling Products” also offering this feature. In case the threat remains applicable, the rating related to the protection of the open sample (presented in the list above) has to be adapted by also considering the availability of a “Sibling Product”. In cases where the availability of the “Sibling Product” would be ranked as public but it is not public knowledge that the “Sibling Product” can be used as a substitute of the used open sample then a rating of ‘Restricted’ is applied instead to cover the effort of identifying the “Sibling Product” and of using it as a substitute to the open sample.

4.7.6 Good usage of open samples and guidance for correct rating

As the privileged usage of open samples / samples with known secrets can be very efficient to speed up evaluations, it also introduces pitfalls that must be avoided. The examples below provide some advice and good practices to correctly require and use open samples / samples with known secrets.

General remarks

As mentioned previously, the goal of the factor open sample / sample with known secrets is to allow an efficient and meaningful evaluation in a maintainable time. The ITSEF must provide a motivated request to the developer explaining the purpose of the open samples / samples with known secrets with respect to the practical tests that will be done on the TOE. This includes a description of the attack path without using open samples / samples with known secrets as well as the estimations of the two different rankings. If an agreement between the Certification Body, the ITSEF and the developer is achieved, the developer will ask the developer of the open sample to provide the open samples / samples with known secrets to the ITSEF. Asking for open samples / samples with known secrets systematically without having in mind the setup of a potential attack derived from the vulnerability analysis is not considered a good practice and shall be declined.

ITSEF should also clearly distinguish between the advantages of using the open sample during an ongoing evaluation from the transferable advantages obtained during other evaluations. If the transferable advantages were gained using open samples from other evaluations, then the rating of this sample also has to be transferred and shall be used in the attack ranking. This is important, especially in the case of evaluations of similar products.

Synchronization on specific operations of interest

Vulnerability analysis requires precise synchronization. This is usually much easier to achieve with an open sample that allows the execution of dedicated code. The open sample could provide some specific trigger signals to indicate e.g. the start and/or end of the operation of interest.

In case of a HW-TOE evaluation firmware modification, or other changes of internal configurations, deviating from the normal product configuration may allow additional synchronization features. Here, the open sample could also be a sample where certain power-saving features (e.g. dynamic Voltage-Frequency scaling), that make some test runs unusable due to unpredictable timing behavior, or performance enhancements (e.g. CPU caches) have been deactivated.

Activation of an available internal interface

Some TOEs have a special interface that may reveal internal data for validation purposes. If this interface is already available and accessible without HW modification, the developer could authorize the ITSEF to use such data in the vulnerability analysis. For example, this could be a sample that exposes the TRNG interface for entropy testing. Here, the interface, which is normally used for validation, and also necessary for TRNG entropy assessment, is used to observe loss of entropy more directly than usually possible. This kind of sample is then rated as open sample.

Pitfalls on attack rankings with and without open samples

Consider here the example of fault attacks where the evaluator has the possibility to find weak spots thanks to open samples, the rating of the attack with and without open samples must be calculated.

Without open samples, the attack on the TOE (combination of platform + application) might not be realistic and might be unfeasible as the number of TOEs needed during the identification phase might reach the ‘Not practical’ ranking. It

is really important to carefully evaluate and not to minimize the influence of the usage of open samples on each factor. Otherwise the usage of open samples would lead to an unjustified rating and in the extreme to a fail of the product if the ranking without open sample is not correctly and fairly done.

Procedure to rank supervised learning attacks using open samples
In case of profiled or supervised learning attacks such as template attacks, supervised machine learning or supervised deep learning approaches it is necessary to possess a sample where the secret information, that is intended to be learnt, can be set to arbitrary values or is known. If this can only be achieved using an open sample the procedure described at the beginning of section 4.7.5 has to be followed (especially as these kinds of attacks are not practical if the learning phase of the attack cannot be applied).

Additionally, there exists the threat that the learning phase could be conducted on a different product and used to attack the TOE. Therefore, the wording of a "Sibling Product" and a procedure that was introduced above has to be followed to address this threat.

Considerations on loading test applications in 'Open platform' evaluations
During 'Open platform' evaluations, loading test applets can help the ITSEF to validate quickly, deeply or with more accuracy the robustness of some specific features. For that purpose, loading capabilities are given to the ITSEF. The points attributed for this advantage need to be considered when the full attack path is rated.

Please note that the knowledge of one loading key of an 'Open platform' will not allow to load applets on all products based on the same platform. Usually more than one set of loading keys exists and may be distributed to different vendors.

For the full attack path rating, either the loading mechanism is broken – and effort to break the loader must be rated in the full attack path – or one of the loading key sets must have been compromised to the attacker. In that case 'open samples / samples with known secrets' points shall be given.

The number of points will be rated according to 'open samples / samples with known secrets' definitions and depends on the protection effort of the loading keys of the platform under evaluation that must be defined by some additional rules provided in the Security Target or in a related security document such as guidance. For example, if no statement is provided about the loading keys management in the ST or in product guidance the ITSEF will consider the minimum criteria factor (Public). If the developer is selling exactly the same platform on different products with different levels of loading keys protections the same rules as for samples with known secrets must be applied.

Obviously, if the loading keys are used by the ITSEF to allow the loading of an applet only providing functional interfaces towards services and that does not provide any significant advantage for the attack realization (no change in factor categories inducing a rating change, e.g. interval change for Elapsed time) no points will be given.

4.8 Calculation of attack potential

Table 13 identifies the factors discussed in the previous sections and associates numeric values with the two aspects of identifying and exploiting a vulnerability. It replaces the equivalent table of the CEM for products that fall under the technical domain of "Smart Cards and similar devices".

Table 13: Final table for the rating factors

Factors	Identification	Exploitation
Elapsed time		
< one hour	0	0
< one day	1	3
< one week	2	4
< one month	3	6
> one month	5	8
> four months[8]	6	10
Not practical	*	*

[8] See Section 4.2 for applicability of this factor.

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
Critical+	9	*
Not practical	*	*
Access to the TOE (1)		
< 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 (2)	3	4
Bespoke	5	6
Multiple Bespoke	7	8
Open samples / Samples with known secrets		
Public/Not required	0	NA
Restricted	2	NA
Sensitive	5	NA
Critical	9	NA
Not practical (Samples with known secrets only)	*	NA

(1) If the package has been claimed as being part or contributing to the TOE security, then extra points to the category 'Access to TOE' may be given as described in Section 4.5.1and in Table 7.

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

* 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.

The following table replaces the equivalent table of the CEM and should be used to obtain a rating for the vulnerability.

Table 14: Rating of vulnerabilites and TOE resistance

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

*final attack potential = identification + exploitation.

5 EXAMPLES OF ATTACK METHODS

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, ITSEFs, Certification Bodies, service providers).

The collection represents the current state of the art at the time. As state of the art is not static this document is reviewed and updated as necessary.

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.

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

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.

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

5.1.2 Impact on TOE

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).

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

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

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.

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.

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.

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

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).

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

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.

The attacks aim at protecting mechanisms and typically include the following:

- Reducing the strength of cryptographic operations,
- Tampering with memory protection mechanisms,
- Affecting non-volatile monotonic counter values.

Creating faults can be used to recover keys or plaintext, to change the results of validation such as authentication or lifecycle state checks, to change the program flow, to provide unauthorized access to protected memory or to enable rollback and replay attacks.

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

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

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, including auxiliary CPU and MPU registers. The modified value is effective until it is overwritten by a new value, and could therefore be used to influence the processing results or the security policy of the device.
- Modifying non-volatile monotonic counter values used to ensure the data freshness. Glitching or reducing the voltage of the power supply at the counter increment can result in a marginal value giving a possibility to roll the counter back, thus enabling replay attacks (if no special countermeasures, like a checksum of a counter, are implemented).
- 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

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).

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

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.

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.

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.

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

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.

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.

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

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. The following paragraphs provide an insight on the various types of currently existing attacks that can be led based on the information leaked through these channels.

For the sake of simplicity, Side-Channel Attacks can be broken down into two main categories:

- In non-profiled SCA (SPA, DPA, CPA, ...), the adversary chooses a priori several attack parameters (target intermediate value, leakage model, ...), using his experience and knowledge of the target (e.g. classical leakage models are Hamming weight and Hamming distance). Then he uses the chosen leakage model for the attack phase.
- In profiled SCA (Template, Stochastic, Machine Learning Attacks, …), the adversary chooses a priori several attack parameters (target intermediate value, ...), but he builds the leakage model a posteriori, during the first step of the attack, called the profiling phase. Thus he requires a copy of the target where it is possible to use chosen or at least known secrets, in order to perform measurements to compute the leakage model. Then he uses the built leakage model for the attack phase.

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.

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).

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.

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. Other side channels, such as timing differences from cache hit/miss arising from speculative/out of order execution (for example Spectre and Meltdown), can also provide useful measurements for this type of attacks. They are also called micro-architectural side-channel attacks.

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.

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.

Recent Machine Learning based attacks exploit the same discriminating criteria (i.e. the dependence between the sensitive data and some statistical moments of the leakage) as a template attack. However, two major differences between these attacks exist:

- The Template attack approximates the data distribution by a multivariate Gaussian distribution (aka Gaussian leakage assumption) whose parameters (i.e. the mean vector and the covariance matrix) are estimated during the profiling phase. This implies that leakage distributions which do not fit in this model make the attack sub-optimal or even ineffective in some contexts.

- The Machine Learning-based attacks make no assumption on the data distribution and build classifications directly from the raw data sets.

These profiled side-channel attacks based on Machine Learning techniques can defeat protected cryptographic implementations resistant to non-profiled SCA including higher-order DPA

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

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.

Typically, 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

A DPA, Template or Machine Learning 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 such means, and obtain the rest by brute-force methods.

For example, 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.

Contrary to higher-order DPA, profiled side-channel attacks based on Machine Learning techniques are particularly suited to attack public key (asymmetric) algorithms such as RSA. In the identification phase, an Artificial Neural Network (ANN) is trained on modular blocks in order to distinguish exponentiation operations in the exploitation phase using a “one trace” matching methodology.

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

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

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

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

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) .

A RNG considered here can be one of the following types[9]:

- 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.

The applicable attack methods vary according to the Type of RNG.

A true RNG may be attacked by[10]:

- permanent or transient influence of the operating conditions (e.g. voltage, frequency, temperature, light)
- non invasive exploitation of signal leakage (e.g. signal on external electrical interfaces)
- physical manipulation of the circuitry (stop the operation, force the line level, modify and/or clone the behaviour, disconnect entropy source)
- wire tapping internal signals (compromise internal states)

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)

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

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.

[9] In the context of smart cards the RNG based on some measurements of environment are not considered to be relevant.
[10] It is here assumed that the direct attack on a true RNG (i.e. guessing the value) is not feasible for any attacker.

5.7.2 Impact on TOE

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.

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.

In case of TRNGs, sufficient tests should be performed (either by the chip platform itself or by the software developer). AIS31[11] 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 section.

5.8 Software Attacks

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.

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.

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).

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.

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.

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.

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
- Software remote attack leveraging TOE's surrounding resources

Attacks related to application isolation (loading, firewalls, etc.) are not described in this section but in section 5.999.

The attacks are of a logical nature, to perform such attacks, it is necessary to have:

[11] Functionality classes and evaluation methodology for physical random number generator, Version 3, 15.05.2013 (BSI).

- 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

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)

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.

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

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).

5.8.1 General description

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.

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.

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).

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).

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.

5.8.2 Information gathering on commands

5.8.2.1 Overview

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.

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[12]).

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.

5.8.2.2 Attack Step Descriptions

Observing Message Sequences

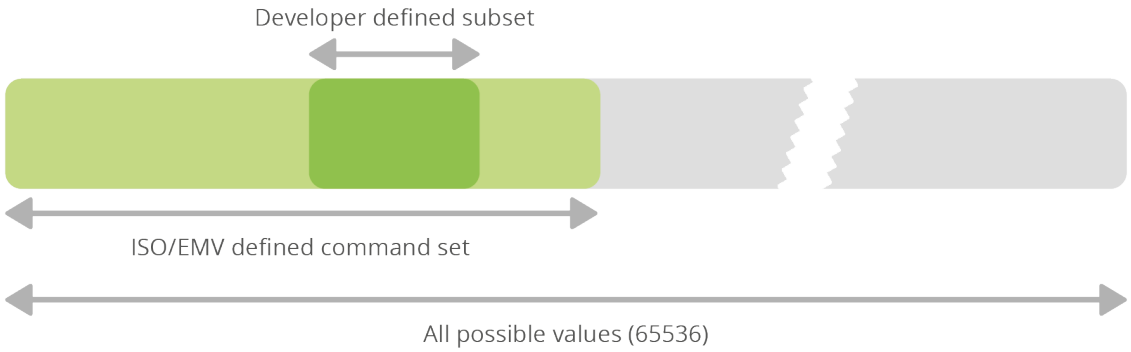
Observing message sequences may result in:

- 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).

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".

Command searches

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.



[12]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.

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

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.

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).

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.

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.

Editing commands

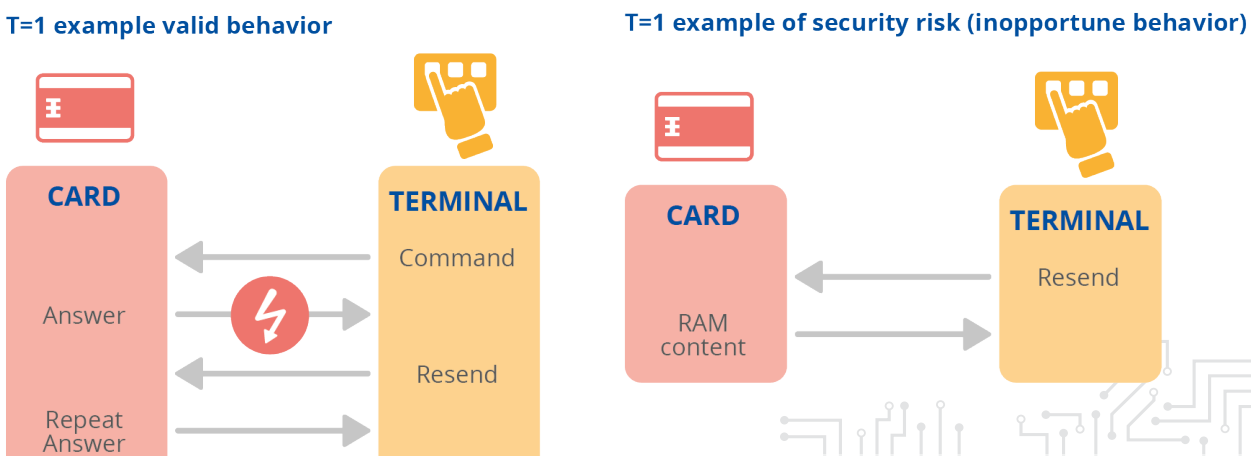
Editing commands is an attack step where the attacker tries to modify commands during the communication sequence to see if the card gives an unexpected reply (these commands may be in an interface specification, or they may have been discovered by observing message sequences or a command search as described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

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.

5.8.3 Direct protocol attacks

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.



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.

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).

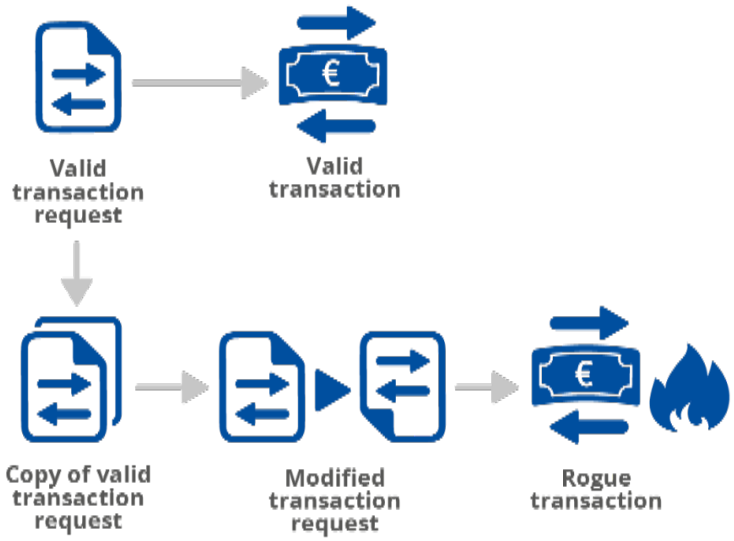
5.8.4 Man-in-the-middle and Replay attacks

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

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).

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[13]). 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.



The picture shows a situation where the attacker copies a valid transaction request, modifies it and sends a second request using the same (or slightly modified) versions of the messages. In general this type of attack might allow the

[13] 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.

attacker to get unauthorised access to a user's assets, for example a bank withdrawal or access to protected system resources.

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).

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

5.8.5 Buffer overflow or stack overflow

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.

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).

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.

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.

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.

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

5.8.6 Communication interface switching

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

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.

5.9 Application isolation

5.9.1 Introduction

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

The 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.

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

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

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

5.9.2 Partial attacks

There is an existing set of technologies applicable to ensure the isolation of applications. These technologies are usually specified in standards, and can be combined in smart card devices.

When performing a full attack, an attacker may need to defeat one or a combination of these technologies. The term partial attacks is used here to describe attacks that have to be combined in the performance of a full attack.

5.9.3 GlobalPlatform partial attacks

The GlobalPlatform (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.

5.9.3.1 Description of a partial attacks

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

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

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

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

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.

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.

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^{14}). If the attack fails (which generally implies a power off), the sequence shall restart from the initial INSTALL [for load] command.

5.9.3.2 Impact on TOE

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

5.9.4 Bytecode verifier partial attacks

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.

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.

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.

5.9.4.1 Description of a partial attack example

Basic type confusion attacks modify the reference of an object by the reference of another object. For instance, we can assign the address of a byte array to a short array in order to dump memory located after the byte array. 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.

[14] 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.

- 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 5.9.3 “GlobalPlatform partial attacks” in order to bypass the loading mechanism.

5.9.4.2 Impact on TOE

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

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.

5.9.5 Defensive virtual machine partial attacks

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[15] 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.

5.9.5.1 Description of a partial attack example

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

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

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.

5.9.5.2 Impact on TOE

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

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.

5.9.6 Firewall partial attacks

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.

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

The Java Card firewall is introduced to provide the sandbox environment for applets running in the same virtual machine.

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

5.9.6.1 Description of partial attacks

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.

5.9.6.2 Impact on TOE

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

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.

5.9.7 Multos partial attacks

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.

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.

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 is why an application cannot access to another application space and so cannot be accessed by other applications.
- The Application loaded on card can contain:
    - MEL instructions
    - Data
    - DIR record: information about the name of the application when loaded on the card
    - 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

5.9.7.1 Description of partial attack

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

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

5.9.7.2 Impact on TOE

The main impacts of this attack are:

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

5.9.8 Full attack path

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

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

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 “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.

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.

5.9.9.1 Impact on TOE

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.

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

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

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 “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[16].

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.

5.9.10.1 Impact on TOE

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

[16] 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.

APPENDIX A

A.1 Access to TOE factor with respect to package removal

It is the developer's choice to include or not to include the package (including the integration structure and overall form factor, e.g. stacked die) in the TOE and describe this as such in the Security Target knowing that this would also impact the ALC_DVS class.

Rating for the removal or preparation of the package is expressed in this document in the 'Access to TOE' factor (see Section 4.5.1) depending on the effort needed to remove the package. Indeed, the package can be seen as a barrier that prevents an attacker from accessing the TOE to perform physical or invasive attacks.

Details of the three rating levels (Low, Medium, and High preparation effort) are defined in the following section. Please note that package preparation methods and the corresponding assessment difficulty are not seen as mature as other areas like SCA or FI. Therefore, the content of this appendix will be revised if needed.

A.2 Effort levels for TOE package preparation effort

This section describes the current view of the effort required for package removal in more detail.

Low preparation effort:
Simple packages that require low preparation effort and that can be removed by standard chemical etching, mechanical action, re-wiring, or similar for the attack path e.g.:

- Conventional smart cards in most cases,
- Standard plastics like DIP, SOIC, QFP, QFN,..., BGA (targeting the more accessible side, typically back side for flip chip).

Medium preparation effort:
Packages that require medium preparation effort and that have a relatively high risk for fatal damage of the TOE (losing the functionality that is target or required for the evaluation) because of special constructions such as:

- Complex package-on-package with glued interposer board,
- Packages with a passive mesh or obstructive wire bonding: This means that the bonding wires are hard to remove/circumvent or hard to re-wire. For example, it requires significant manual reverse-engineering (>1 week) of several hundred pins to be obtained by generating a bonding map. And, effort to translate the bonding map to a bonding machine file format, for the use of an automated bonding machine. Note that if the reverse-engineering does not need to be redone in exploitation, consequently points in exploitation shall only be given if the remaining attack path still requires specialized equipment or above.
  Separation of the dies can be difficult as there are some functional dependencies in place between the dies, which have to be reconnected involving some reverse engineering, development of a testing device between the dies, and adds risk of putting TOE out of operation,
- Casting compounds, for example based on synthetic material like resin, which require material-specific chemistry, as mechanical removal of the package leads to fatal damage of the TOE with high probability. Such casting compounds are for example used in HSMs (key generation devices in Trust Centres). However, the material specification and state-of-the-art removal methods shall then be subject of the evaluation. The ITSEF should try to remove the package using standard chemical methods, for example a wave of hot fuming sulfuric acid, application of fluoric acid and other.

Note: At this point in time there is no harmonisation between the countries regarding package removal methods and therefore it is a case by case decision/discussion with Certification Body to define what is standard.

High preparation effort:
Packages that require high preparation effort, multiple experts and rare bespoke tooling, which are not claimed as security functionality, such as:

- Chip-on-chip with critical functional dependency that require a wing board to be able to work: It is important to consider methods to circumvent dependencies, e.g. run external memory with lower frequency and

similar. In this example the TOE is not functional without external memory, or the TOE checks presence of the memory, but the SoC adds no protection means for the TOE.
If there are TOE checks for external components, then circumventing these checks matters the evaluation,

- Packages with an active mesh meaning for example that the mesh is connected to the TOE and monitored by the TOE for damages,
- There could be casting compounds, for example on ceramic basis, which cannot be removed without fatal damage of the TOE by using mechanical and also standard chemistry. The removal is therefore either not practical to state-of-the-art-knowledge, as outlined below, or subject to bespoke methods known to the vendor only and shared with the ITSEF in the course of evaluation. I.e. there should be no publicly known method to remove the package material easily or with state-of-the-art chemical methods. For that reason, the material specification and check of the state-of-the-art removal methods shall be subject of the evaluation, which may involve external experts from other faculty, such as chemistry and other. The vendor can also be required to provide material samples for the chemical analysis and attempts. Those material samples can but must not include the TOE.

A.3 Examples for rating the removal of packages

The following package descriptions are not based on existing products and are provided only as examples for the rating methodology:

Example 1

This is the baseline situation where the package does not contribute to the attack resistance. The example assumes a simple Light Fault Injection (e.g. authentication bypass) on a vulnerable TOE in Chip Scale Package or bare die. In this case the effort and skills needed to prepare the TOE for LFI are very limited because CSPs can be easily dissolved by chemical etching. For bare dies no preparation at all is needed.

Table 15: Combined basic laser fault injection and package removal rating for low package preparation effort

LFI on CSP or bare die			
Factors	**Description**	**Identification**	**Exploitation**
Elapsed time	For setting up the equipment and performing the attack an attacker would spend more than a week but less than a month.	< one month (3)	< one week (4)
Expertise	Only proficient knowledge is required with respect to the functionality under attack.	Proficient (2)	Proficient (2)
Knowledge of the TOE	The attack can be performed using public domain information.	Public (0)	Public (0)
Access to TOE	There is no risk the TOE is damaged during opening. So, no points will be given here.	<10 samples (0) **Low preparation effort +(0)**	<10 samples (0) **Low preparation effort +(0)**
Equipment	Just a standard LFI setup is required. No equipment for package removal.	Specialized (3)	Specialized (4)
Open samples		not required (0)	-
Subtotal		8	10
Total		**18 (Basic)**	

Example 2

Light Fault Injection on a vulnerable TOE in a package-on-package configuration. This is the same TOE as Example 1, but in a different type of package. The vendor claims that the package adds extra security to the TOE. The TOE (bottom package) can work independently from the supporting device inside the top package. The die inside the bottom package is sandwiched between the lower carrier board and upper carrier board. The upper board completely covers the TOE. The voids between the carrier boards are filled with resin. Opening this package requires substantially more time and tools compared to CSPs and bare dies. First the top package must be removed. Then an opening must be made in the upper carrier board without damaging the SoC die inside the bottom package. The SoC die must be exposed for LFI preparation, which requires etching. Finally, the TOE must be fitted inside an LFI set-up and the attack must be performed.

Table 16: Combined basic laser fault injection and package removal rating for medium package preparation effort

LFI on package-on-package without dependency on supporting device (bold text represents the extra resistance provided by package)			
Factors	**Description**	**Identification**	**Exploitation**
Elapsed time	For setting up the equipment and performing the attack an attacker would spend more than a week but less than a month.	< one month (3)	< one week (4)
Expertise	Only proficient knowledge is required with respect to the functionality under attack.	Proficient (2)	Proficient (2)
Knowledge of the TOE	The attack can be performed using public domain information.	Public (0)	Public (0)
Access to TOE	There most likely will be some TOEs destroyed during separation and opening, but not likely more than 10 before successful preparation. **Based on the above de-packaging description it is considered as difficult to prepare (medium effort) the SoC to perform the attack.**	<10 samples (0) **Medium preparation effort +(1)**	<10 samples (0) **Medium preparation effort +(2)**
Equipment	Just a standard LFI setup is required.	Specialized (3)	Specialized (4)
Open samples		not required (0)	-
Subtotal		9	12
Total		**21 (Enhanced basic)**	

Example 3

Light Fault Injection on a vulnerable TOE in a chip-on-chip package configuration with high data rate interconnections. This is the same TOE as Example 1, but again in a different type of package. These high-speed connections require critical routing to guarantee signal integrity. The chips are glued together which makes separation without damage extremely difficult. The pitch between the contacts is small. Once separated an interface board is required to re-connect both chips. Connecting the interface board to both chips by means of wire bonding is not trivial due to the fine pitch and routing requirements. Further assumptions: Security claim on the package by the developer, upper board completely covers the TOE.

Table 17: Combined basic laser fault injection and package removal rating for high package preparation effort

LFI on chip-on-chip package with dependency on supporting chip (bold text represents the extra resistance provided by package)			
Factors	**Description**	**Identification**	**Exploitation**
Elapsed time	For setting up the equipment and performing the attack an attacker would spend more than a week but less than a month.	< one month (3)	< one week (4)
Expertise	Only proficient knowledge is required with respect to the functionality under attack.	Proficient (2)	Proficient (2)
Knowledge of the TOE	The attack can be performed using public domain information.	Public (0)	Public (0)
Access to TOE	**There most likely will be many TOEs destroyed during separation and opening, especially while figuring out the best approach.** **Based on the de-packaging description it is considered as hard to prepare (high preparation effort) the SoC to perform the attack.**	<10 samples (0) **High preparation effort +(2)**	<10 samples (0) **High preparation effort +(4)**
Equipment	A standard LFI set-up is required.	Specialized (3)	Specialized (4)
Open samples		not required (0)	-
Subtotal		10	14
Total		**24 (Enhanced Basic)**	

ABOUT ENISA

The mission of the European Union Agency for Cybersecurity (ENISA) is to achieve a high common level of cybersecurity across the Union, by actively supporting Member States, Union institutions, bodies, offices and agencies in improving cybersecurity. We contribute to policy development and implementation, support capacity building and preparedness, facilitate operational cooperation at Union level, enhance the trustworthiness of ICT products, services and processes by rolling out cybersecurity certification schemes, enable knowledge sharing, research, innovation and awareness building, whilst developing cross-border communities. Our goal is to strengthen trust in the connected economy, boost resilience of the Union's infrastructure and services and keep our society cyber secure. More information about ENISA and its work can be found at www.enisa.europa.eu.

ENISA
European Union Agency for Cybersecurity

Athens Office
1 Vasilissis Sofias Str
151 24 Marousi, Attiki, Greece

Heraklion office
95 Nikolaou Plastira
700 13 Vassilika Vouton, Heraklion, Greece

enisa.europa.eu