



Joint Interpretation Library

Security Architecture requirements (ADV_ARC) for smart cards and similar devices

Document purpose: provide guidance to developers to fulfill the Security Architecture requirements of CC V3 ADV_ARC family.

Version 1.0 (for trial use)
June 2008

This page is intentionally left blank

Table of contents

1	Introduction	5
1.1	Background.....	5
1.2	Objective of the document	5
2	Essence of the Security Architecture.....	6
2.1	TSF “always invoked” – non-bypassability	6
2.2	Protection of TSF – Secure Initialisation and Self-protection	7
2.2.1	Secure start-up	7
2.2.2	Self-protection	8
2.3	Security domains	9
2.3.1	Definition of security domain	9
2.4	Level of description in ADV_ARC	10
3	Security Architecture content of presentation example.....	11

Appendix 1: Security Architecture (ADV_ARC) example

1	Introduction	13
1.1	Purpose and Scope	13
1.2	Documentation.....	13
1.3	TOE Summary Specification as defined in the ST	13
2	Security domain separation	14
2.1	Security domains	14
3	Initialization / start-up	15
4	Global security architecture.....	16
4.1	Example for Security IC	16
4.1.1	Security Services	16
4.1.2	Security Mechanisms	17

4.2	Example for smart security devices	18
4.2.1	Security services.....	18
4.2.2	Security mechanisms.....	18
5	Cooperation of Security Mechanisms	20
5.1	Mapping of SFR's, Security Features and Security Mechanisms	20
5.2	Self-protection and Non-Bypassability.....	21
5.2.1	Probing.....	21
5.2.2	Manipulation	22
5.2.3	SPA/DPA attacks.....	23
5.2.4	Timing attacks	23
5.2.5	Environmental Stress (V, f, T).....	23
5.2.6	Light Attacks (Frontside, Backside)	23
5.2.7	<Other fault injection attacks>.....	23
5.2.8	Re-activate IC Test mode	24
5.2.9	Manipulate/Modify rights for restricted modes of operation	24
5.2.10	<Other attacks>	25
6	Bibliography	26

1 Introduction

1.1 Background

The version 3 of Common Criteria (CC) introduces a new security assurance requirements (SAR) family Security Architecture (ADV_ARC). Its objective is described in paragraph 214 of CC part 3 as follows:

“The objective of this family is for the developer to provide a description of the security architecture of the TSF. This will allow analysis of the information that, when coupled with the other evidence presented for the TSF, will confirm the TSF achieves the desired properties. The security architecture descriptions support the implicit claim that security analysis of the TOE can be achieved by examining the TSF; without a sound architecture, the entire TOE functionality would have to be examined.”

The family ADV_ARC requires the TOE security architecture to describe the self-protection, domain separation and non-bypassability principles. The Security Architecture shall also describe the secure TOE security functionality (TSF) initialisation.

These properties are distinct from security functionality expressed by CC part 2 security functional requirements (SFR) because they largely have no directly observable interface at the TOE Security Functionality (TSF).

In fact the SFR describe the security functionality provided to the users for protection of their assets from unauthorised disclosure, modification, or loss of use. The SFR are defined in the security target (cf. to family ASE_REQ). The TSF interfaces are described in the functional specification (cf. to family ADV_FSP) and refined on level of sub-systems and modules in the TOE design description (cf. to family ADV_TDS).

The properties of self-protection, domain separation, non-bypassability and secure initialisation are achieved through the correct TOE design implementation. A sound security architecture is therefore necessary to ensure the effectiveness of the TSF to enforce all its SFRs.

The smart card technology combines security integrated circuits, operating systems and applications to high secure devices.

1.2 Objective of the document

The current document provides guidance for the developer and for the evaluator on how to apply the assurance requirements of the family ADV_ARC to smart cards and similar devices.

The smart card technology requires special interpretation because it combines security integrated circuits, operating systems and applications to high secure devices.

2 Essence of the Security Architecture

ADV_ARC.1.3D requires the developer to provide a security architecture description of the TSF. The security architecture description addresses properties and architecture-oriented features of the TOE. They contribute to security by securing directly the TSF and protecting the user assets through the enforced SFR of the TSF.

The functional requirement class FPT: Protection of TSF contains families of functional requirements that relate to the integrity and management of the mechanisms that constitute the TSF and to the integrity of TSF data. Components from the class FPT: Protection of the TSF class is necessary to provide requirements that the SFPs in the TOE cannot be tampered with or bypassed. Important SFR families are Testing of external entities (FPT_TEE)¹, TSF physical protection (FPT_PHP), TSF self test (FPT_TST) and Fail secure (FPT_FLS).

Some features of the security architecture in CC version 3 were described as security functional requirements in CC version 2: non-bypassability was described by the SFR family Reference mediation (FPT_RVM) and domain separation by the SFR family Domain separation (FPT_SEP). When appropriate components from the families FPT_SEP and FPT_RVM were combined with the appropriate components from TSF internals (ADV_INT), the TOE can be said to have what has been traditionally called a “Reference Monitor” (cf. CC version 2.3, part 2, chapter 6 and annex J). As the families FPT_SEP and FPT_RVM are removed from CC part 2, the related security features shall now be described through ADV_ARC family.

2.1 TSF “always invoked” – non-bypassability

The component ADV_ARC.1.1D requires the developer to design and implement the TOE so that the security features of the TSF cannot be bypassed. Furthermore the security architecture description shall **demonstrate** that the TSF prevents bypass of the SFR-enforcing functionality (cf. to ADV_ARC.1.5C).

Non-bypassability is a property that the security functionality as specified by the SFRs is always invoked and cannot be circumvented when appropriate for that specific mechanism (cf. to Annex A of CC part 3, paragraph 517).

The CC part 3, A.1.2.3 discusses the bypassability of the SFR-enforcement through different interfaces and the information provided in the functional specification (ADV_FSP) and TOE design (ADV_TDS) about the operations available through these interfaces.

TSFI - consist of all means for users

- (1) to invoke a service from the TSF (by supplying data that is processed by the TSF) and the corresponding responses to those service invocations or
- (2) to affect the behaviour or the security of the TSF.

¹ Testing of external entities (FPT_TEE) defines requirements for the TSF to perform testing to demonstrate the security assumptions made about the ‘underneath’ platform upon which the TSF relies. In composite evaluation the platform is normally part of the TOE thus FPT_TEE is not used for smart cards.

Port - a physical entry or exit point of the TOE that provides access to the TOE for physical signals, represented by logical interfaces, including power supply.

Therefore a port may provide more information than necessary for the TSFI. This additional information may bypass the security functionality intentionally provided through this interface (e.g. by a side channel).

Interface of physical protection (FPT_PHP) is the TSF boundary.

TOE physical boundary - an explicitly defined continuous perimeter that establishes the physical bounds of the TOE and contains all the hardware, software, and/or firmware components of the TOE.

Physical boundary shall be described and examined for

- Ø physical protection (FPT_PHP)
- Ø completeness of the port and interface description, including areas of emanation and for irradiation
- Ø consistency of TOE architecture, design, and implementation

As stated above, the developer shall consider the SFR_enforcing functionalities and explain how the different ‘design elements’/mechanisms cooperate and contribute to prevent by-passing of these SFR_enforcing functionalities. Demonstration here means that developer has to provide an explanation on how SFR_enforcing functionalities cannot be by-passed.

2.2 Protection of TSF – Secure Initialisation and Self-protection

2.2.1 Secure start-up

The security architecture description shall describe how the TSF initialization process is secure (cf. ADV_ARC.1.3C). The information provided in the security architecture description relating to TSF initialisation is directed at the TOE components that are involved in bringing the TSF from the “down” state (e.g. power-off) into an initial secure state (i.e. when all parts of the TSF are operational) (cf. CEM paragraph 529).

The TSF may have parts providing their security function when the TOE is not running. E.g. the physical protection of a security integrated circuit shall resist tampering attacks according to FPT_PHP.3 even if power is off.

Other parts of the TSF may be activated during start-up of the TOE before or at the same time when the relevant TOE functionality is activated. E.g. sensors shall control the environmental conditions for the normal secure operation of a security IC at time the operating system of the smart card starts-up. The operating system shall check the integrity of stored TSF data before relying on it.

For smart cards the secure initialisation process of the TSF covers power-on start-up, entering and wake-up from power save modes or any kind of reset.

Secure initialisation of the TSF is a prerequisite for non-bypassability of the TSF. The TSF controlling other TOE functions shall be activated before these functions are available by possibly untrusted entities. The TSF initialisation process shall be synchronised with the start-up of the TOE.

2.2.2 Self-protection

The component ADV_ARC.1.2D requires the developer to design and implement the TSF so that it is able to protect itself from tampering by untrusted active entities. Furthermore the security architecture description shall demonstrate that the TSF protects itself from tampering (cf. to ADV_ARC.1.4C).

Self-protection refers to the ability of the TSF to protect itself from manipulation from external entities that may result in changes to the TSF, so that it no longer fulfil the SFRs . For the purposes of the security architecture description, the notion of *self-protection* applies only to the services provided by the TSF through its TSFI, and not to services provided by underlying IT entities that it uses. (cf. CEM paragraph 532).

Self-protection of the TSF will be achieved by:

- Ø Self-protection of TSF mechanisms: the ability of a TSF mechanism to protect themselves against direct attacks to interfere, to manipulate or to disable this mechanism.
- Ø Binding of TSF mechanisms: the ability of the TSF mechanisms to work together in a way that is mutually supportive and provides an integrated and effective whole.

Example of self-protection of a TSF mechanism is the protection against physical probing of a security IC to manipulate or to disable TSF features. In some cases the physical protection mechanism must resist such attacks independent on any other TSF mechanism. In other cases the physical protection mechanism detects such attacks and the operating system reacts on it entering a secure state.

Example of binding of TSF mechanisms in case of smart cards is the combination of hardware and software TSF mechanisms

- Ø to ensure the stable correct execution of the embedded software under specified operational conditions,
- Ø to detect errors of the execution caused by perturbation and
- Ø to enter a secure state if detected errors can not be automatically corrected.

Note that buffer overflows are attack techniques where the passive external entity becomes an active internal entity because the TOE executes the provided input data as code.

Though single mechanisms may be required by SFR (like error detection by FPT_TST.1 and secure state by FPT_FLS.1) the security architecture description shall demonstrate why these mechanisms as a whole counter the attack (e.g. how the error will be detected and the secure state was reached even if perturbation of the execution of a command was successful).

2.3 Security domains

2.3.1 Definition of security domain

The security architecture description shall describe the security domains maintained by the TSF consistently with the SFRs (cf. to ADV_ARC.1.2C).

Domain separation is a property whereby the TSF creates separate *security domains* on its own and for each untrusted active entity to operate on its resources, and then keeps those domains separated from one another so that no entity can run in the domain of any other (cf. CC part 3, paragraph 515, 524 and 578).

Security domains refer to environments supplied by the TSF for use by potentially-harmful entities (cf. CEM, paragraph 527). The environment provided by the TSF to an entity may comprise resources

- ∅ for input and output to interact with users,
- ∅ address space to access operational memory or functional registers,
- ∅ commands available for execution by an entity,
- ∅ per-process environment variables,
- ∅ stored data.

External entities provide input data and receive output data to interact with the TOE through the TOE interfaces. This interaction between users and the TOE is controlled by the TSF to enforce the SFRs. An internal entity (process) may use the provided resources within its environment according their own security policy (outside control of the TSF). That is why this environment is called security domain.

The TSF runs its own security domain or several security domains. If the TSF provides security domains for other active entities the TSF shall protect their own domain(s) against adversary actions of these potentially-harmful entities on TSF resources. Moreover, the TSF keeps those domains separated from one another so that no entity can run in the domain of any other.

An example of a security domain is the “sandbox” provided by the java cards for the execution of an applet. The Java run time environment (JRE) interprets the applet code by operating on well defined resources of this applet and providing services for this applet (e.g. cryptographic functions). The TSF of the java card controls access to these services and the communication between applets by firewall mechanisms. But the TSF does not control the operation on objects within the sandbox.

The TSF may not provide any security domain for other active entities if the TSF is the only active entity and the TOE interacts with external entity through the TSFI only. In this case all of the interactions available to users are severely constrained by the TSF reading and writing the IO buffer.. The security architecture will describe that there is no security domain available for active entities (except the TSF itself). A firewall may be seen as one example of such TOE (cf. to example in paragraph 528 of CEM).

The security domains defined in ADV_ARC **should be under SFR control** (e.g FDP_ACC: access control). If a security domain is defined and not controlled by a SFR, that would mean that a SFR is missing.

2.4 Level of description in ADV_ARC

ADV_ARC.1.1C requires the architecture description to be “at a level of detail commensurate with the description of the SFR-enforcing abstractions described in the TOE design document”. This means that security architecture argumentation shall be at the most meaningful level, and the level of description provided in ARC shall be the level used for the description of SFR_Enforcing in ADV_TDS: Subsystems or modules.

3 Security Architecture content of presentation example

An example of Security Architecture document content for Smart Cards and similar devices is presented in Appendix 1.

This template which has been elaborated by Smart Cards evaluation experts fulfills ADV_ARC.1 requirements. The evidences to be provided are similar to CC 2.3 AVA_VLA developer's vulnerability analysis.

Appendix 1: Security Architecture (ADV_ARC) example

<Product Name Title>

Security Architecture description

Glossary

Countermeasures	Security Measures and/or Security Features designed to avert the threats defined in the Security Target (e.g. to counter physical tampering, environmental stress and other attacks).
Security Feature	Combination of several Security Mechanisms to counter one or more attacks. Often the composition of a Security Feature only becomes clear when considering a specific attack path during vulnerability analysis.
Security Measure	Technical and organizational measures that ensure the security of the development environment and production environment, therefore Security Measures do pertain to the TOE but are not part of the TOE.
Security Mechanism	Technical implementation of a countermeasure (lowest level of decomposition). The TOE Security Mechanisms need to work together in different combinations to counter attacks. A particular security mechanism may be needed in different security features depending on the attack path.
Security Service	The Security IC provides Security Services for the Security IC Embedded Software (e.g. cryptographic operations, random number generator).
xxx	xxx

Abbreviations

CC	Common Criteria
ST	Security Target
SFR	Security Functional Requirement
SM	Security Mechanism
SF	Security Feature
SS	Security Service
TOE	Target of Evaluation
TSF	TOE Security Functionality
SFR	Security Functional Requirement

1 Introduction

1.1 Purpose and Scope

This document contains the security architecture description for the <Product Name Title> as required by the CC part 3, 12.1 ADV_ARC:

“The security architecture description supports the implicit claim that security analysis of the TOE can be achieved by examining the TSF; without a sound architecture, the entire TOE functionality would have to be examined.”

1.2 Documentation

For information and details beyond this document refer to the following specifications:

- ∅ Functional specification according to CC part 3, ADV_FSP
- ∅ TOE design according to CC part 3, ADV_TDS
- ∅ Implementation representation according to CC part 3, ADV_IMP
- ∅ Others ...

1.3 TOE Summary Specification as defined in the ST

In this section the developer may provide an overview of the TOE Summary Specification (TSS) defined in the Security Target.

The TSS aims providing a general understanding of how the TOE implements the SFRs. As we will see further in this document, especially in the case of Security IC, the Security Functionalities defined in the TSS will contribute to the Security Architecture of the TOE.

Smart card developers often define Security Functionalities, also called Security Services or Security Features as a convenient intermediate step in the mapping to groups of SFRs to describe the TSS.

<Example>

The TOE provides the following TOE Security Services (SS):

- ∅ Random number generator
- ∅ Cryptographic operations
- ∅ Others ...

See chapter 4.1 for more details

The TOE provides the following Security Features (SF):

- ∅ Control of operating conditions (for the Security IC)
- ∅ Key reading method
- ∅ Authentication verification method
- ∅ Others ...

See chapter 4.3 for more details

2 Security domain separation

According to CC part 3, Annex A.1.1, 515 “domain separation is a property whereby the TSF creates separate security domains for each untrusted active entity to operate on its resource, and then keeps those domains separated from one another so that no entity can run in the domain of any other.”

Excerpt from BSI-PP-0035-2007

The Security Architecture shall describe how the security architecture design and implementation prevents bypass of SFR limiting the availability of the Test Features as required by the Limited capability and availability policy defined in FMT_LIM.2. This includes any configuration of the availability of the Test Features performed by the TOE Manufacturer before TOE Delivery.

2.1 Security domains

For Security IC's it can be assumed that Security domains exist called Test mode and Application mode (or similar). Under Security Targets compliant to BSI-PP-0035 2007, these are defined as FMT_LIM.1 and FMT_LIM.2. Implementation of these security domains is considered in other ADV families and is not repeated here.

Other security domains might exist, for example due to an implemented Memory Management Unit. In this case they should be described in the Security Target by Security Functional Requirements (SFR's). If a Security Domain is not explicitly described within the Security Target, it has to be described here.

For the composite smart card product, there are also different life-cycle modes as initialization, personalization or application mode. If these modes involve interfaces and code execution that are part of the TOE and are described in the Security Target by Security Functional Requirements (SFR's) they will not be described in further details as implementation of these security domains is considered in other ADV families. If a Security Domain is not explicitly described within the Security Target, it has to be described here.

Security domains may exist due to the TOE limits. Typical example is a Java Card that reserves security domains for the execution of each application running on the card. If the TOE contains several 'trusted' applications sharing the same TOE resources (e.g cryptographic libraries), the separation of each application own resources is covered by the Security Target SFRs by example, through access control. But, in the case were one or several applications are not in the TOE (not evaluated), this section shall describe how the TOE maintains domains separation (protection from access to TOE 'evaluated' resources).

3 Initialization / start-up

According to CEM v.3.1, 529 “The information provided in the security architecture description relating to TSF initialization is directed at the TOE components that are involved in bringing the TSF into an initial secure state (i.e. when all parts of the TSF are operational) when power-on or a reset is applied. This discussion in the security architecture description should list the system initialization components and the processing that occurs in transitioning from the “down” state to the initial secure state.

Excerpt from BSI-PP0035-2007

The Security Architecture description of the TSF initialization process shall include the procedures to establish full functionality after power-up, state transitions from the secure state as required by FPT_FLS.1 and any state transitions of power save modes if provided by the TOE.

<Example for security IC>

As long as the operating conditions of the device in terms of voltage, frequency and temperature did not reach the specified range the whole IC is kept in reset.

After reset the device starts with the internally functionality test as

(It has to be described why the boot sequence (start up sequence) is implemented such that a secure state will be reached.)

<Example for smart security device >

The TSF discriminate between a cold reset (power-on) and, warm reset (reset).

Before allowing any operation, the TSF executes the following operations and controls to put the TOE in a secure state.

The Developer shall provide here information on the boot sequence and all security features initialized during the start- up (e.g backup recovery, sensitive areas integrity check, fault counters ...) and describe how the initialization process is secure.

4 Global security architecture

In this section the developer shall provide an overview of the security mechanisms that contribute to the non-bypassability and self protection of the TSF. The content presentation will differ when considering the Security IC and the smart security device

The Security IC PP-BSI-0035-2007 SFRs already cover most of the attacks to consider for non by-passing and self-protection. As a consequence, most of the design information to put in ADV_ARC are already in ADV_TDS design documentation. Therefore, description of the Security IC mechanisms shall not be repeated but refer to ADV_TDS documentation.

For the composite smart security device, different Protection Profiles are claimed. The SFRS defined in the PPs may not cover self protection and Non-bypassability (FPT_RVM no longer exist). Therefore, reference to ADV_TDS is not as systematic as in the case of a Security IC.

By example:

- ∅ A mechanism designed to fulfill cryptographic Key access requirement (FCS_CKM.3) may involve specific method to hide the key usage, and therefore contributes to self-protection.
- ∅ A mechanism designed to fulfill unobservability requirement (FPR_UNO.1) may exhibit design properties used for self-protection or non-bypassing of other parts of the TSF.
- ∅ A mechanism specifically design to protect authentication (FIA_AFL.1)

In the first two cases, reference will be made to ADV_TDS, in the third case, the mechanisms will be described in ADV_ARC.

4.1 Example for Security IC

This section gives an overview of all Security Mechanisms of the TOE. How they are linked (-> Security Features) to prevent attacks or at least to make them very difficult is described in chapter 5. Security Mechanisms (SM.x) and Security Features (SF.x) mentioned here are valid for all or major parts of the TOE. The descriptions of the Security Mechanisms are given in detail in the ADV documentation or in the following chapter.

For convenience of the reader and for completeness also the Security Services (SS.x) are listed.

Although probably already described in further ADV documents the relationship (mapping) between SFR's and Security Services and Security Features on one side and the relationship between Security Features and Security Mechanisms are repeated here.

4.1.1 Security Services

SS.1 RNG - Random number Generator

The Random Number Generator is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x.

SS.2 DES algorithm

The DES algorithm is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

SS.3 AES algorithm

The AES algorithm is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

SS.x <crypto algorithm x>

The SS.x <crypto algorithm x> is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

SS.x+1 MMU Memory Management Unit

SS.x+1 <MMU> is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

SS.x+2 <Other Security Services>

(Add further Security Services if provided by the TOE, according to the Security Target.)

4.1.2 Security Mechanisms

This chapter includes a list of the Security Mechanisms of the TOE and a description or a reference to the specific ADV_XXX document which includes the description of all Security Mechanisms. For the convenience of the reader descriptions have been included here where found appropriate.

(The following description serves as an example.)

4.1.2.1 Control of operating conditions

The device has several types of exception sensors monitoring its extrinsic operating parameters as voltage, frequency, temperature, and light, which are totally independent from each other and from the other functionality of the TOE. The exception sensors guarantee the device to be operated under the specified conditions. In case specified operating conditions are not fulfilled the sensors generate an internal device reset.

The sensors can be enabled or disabled in order to allow for testing the device functions beyond the sensor threshold. By testing devices in production outside the sensor limit the correct function of the device and its sensors at the sensor limits is guaranteed. After delivery (Application mode) the hardware does not allow the embedded software to interfere with the sensors.

SM.1 Low and High Frequency Sensors

The Smart Card Controller cannot be operated at low clock frequency ($f_{CLK} < f_{CLK(LFS)}$, LFS=Low Frequency Sensor). At low frequencies a permanent RESET is performed. If the clock frequency raises above the high frequency limit ($f_{CLK} > f_{CLK(HFS)}$, HFS=High Frequency Sensor), reset will be executed as well.

SM.1 is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

SM.2 Voltage Sensors

The Smart Card Controller includes a power-on-reset circuitry controlling the dedicated power-on and power-off sequence.

When the voltage $VDD < VDD_{(LVS)}$ (LVS=Low VDD! Sensor) or $VDDs > VDD_{(HVS)}$ (HVS=High VDDs Sensor) a reset will be executed.

SM.2 is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

SM.3 Temperature Error Sensor

The device supplies a temperature error sensor monitoring the operating temperature. If the temperature drops below $T < T_{(LTS)}$ (LTS=Low Temperature Sensor) or if the temperature raises above $T > T_{(HTS)}$ an internal reset will be executed.

SM.3 is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

SM.4 Light Sensor

There are a number of light sensors located on the chip that are designed in a way that assures normal device functionality under spec conditions, but invokes a sensor reset, if the device is exposed to strong light emitters.

SM.4 is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

4.1.2.2 *Shields*

SM.5 Active Security Routing

(Description of the security routing)

SM.5 is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

4.1.2.3 *<Further Security Mechanisms>*

(add all Security Mechanism of the TOE, consider grouping)

4.1.2.4 *Further Countermeasures*

For functionality, which contribute to the security of the TOE but is not described as a Security Mechanism in the ADV documentation, a detailed description of the functionality of the security mechanism can be given in the following.

(The following description serves as an example.)

SM.n DES calculation protection

The security mechanism calculates DES.detailed description of the implementation

SM.n+1 Shadow Memory for NVM write parameter

The security mechanism stores the address and length information for the NVM write parameterdetailed description of the implementation

4.2 Example for smart security devices

This section gives an overview of all Security Services and Security Mechanisms of the TOE. How they are linked to prevent attacks and protect the TSF is described in chapter 5.

Some of the Security mechanisms are already described in ADV documents. For convenience of the reader, all the security services and mechanisms are listed here

4.2.1 Security services

SS.1 Software RNG - Random number Generator

The Random Number Generator is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x.

SS.2 Software DES algorithm

The DES algorithm is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

SS.3 Software RSA algorithm

The RSA algorithm is described in ADV_FSP chapter x.x and in ADV_TDS chapter x.x

4.2.2 Security mechanisms

This chapter includes a list of the Security Mechanisms of the TOE and a description or a reference to the specific ADV_xxx document which includes the description of all Security Mechanisms. For the convenience of the reader descriptions have been included here where found appropriate.

4.2.2.1 *Authentication protection*

SF.1 Authentication tries mechanism protection

Usage of the TOE is allowed after a successful authentication. The number of authentication attempts is

limited to [Max] value. This mechanism is designed to ensure that the value cannot be modified by an attacker. (Further describe the mechanism)

SF.2 Access control protection

This mechanism is designed to detect abnormal operations that might indicate an attack by an external entity. (Further describe the mechanism)

4.2.2.2 Secret protection mechanisms

SF.3 Key protection mechanism

This mechanism is designed to prevent attempt to disclose the value of the key used during the encryption operation. (Further describe the mechanism)

4.2.2.3 SPA/DPA protection mechanisms

SF.4 Side Channel protection

This mechanism is designed to make difficult the observability of operations and prevent the possibility for an attacker to synchronize code execution with an external signal. Further describe the mechanism)

4.2.2.4 Fault attack protection mechanisms

SF.5 Fault detection

This mechanism is designed to detect abnormal operation execution by passing due to fault attacks. (Further describe the mechanism))

4.2.2.5 <Other Security Mechanisms>

(Add all Security Mechanism of the TOE, consider grouping)

5 Cooperation of Security Mechanisms

5.1 Mapping of SFR's, Security Features and Security Mechanisms

<Note> As explained in the introduction of chapter 4, the SFRS in the Security IC Protection profile, cover already by-passing and self protection. Therefore, the mapping of SFRS, Security Features and Security Mechanisms is quite straight forward from the Security Target.

In the case of a composite smart security device, this mapping may not be relevant as protection profiles address application and user data protection. The architecture provides design elements to protect the TSF (not the SFRs).

The following description serves as an example for a security IC.

In order to fulfil the Security Functional Requirements (SFR), the TOE provides Security Features (SF) and/or Security Services (SS). The SF and/or SS are composed of different Security Mechanisms, which are working together to meet the Security Functional Requirements and to protect the TOE from tampering and bypassing.

The following description shall show how the SS (Security Services) and/or the SF (Security Features) and the SM (Security Mechanisms) are working together to fulfil the Security Functional Requirements of the TOE.

The TOE provides the following Security Services (SS.x) and Security Features (SF.x):

SS.1	Random Number Generation
SS.2	Cryptographic Operation
SS.3	< further Security Services>
.....	
SF.1:	Operating state checking
SF.2:	Phase management with test mode look-out
SF.3:	Protection against snooping
.....	

The Security Services and Security Features of the TOE (SS.1, SF.1 to SF.3) are described in the document ADV_FSP, section 2.

The Table 1 shows that the Security Features and the Security Service of the TOE are covering all the required Security Functional Requirements.

The Table 2 shows the contribution of the security mechanisms (SM.x) to the Security Features (SF.x) and the Security Services (SS.x) of the TOE. The cooperation of the security mechanisms within the Security Features and the Security Services are described in the document ADV_FSP, Chapter x.

	SF.1	SF.2	SF.3	SF.x	SS. 1	SS.x
FAU_SAS.1		X				
FMT_LIM.1		X				
FMT_LIM.2		X				
FPT_FLS.1	X					
FPT_PHP.3			X			

FRU_FLT.2	X					
.....						
FCS_RND.1					X	

Table 1 Mapping of the SFR to the Security Features (SF) and Security Services (SS)

SF and SS Security Mechanisms	SF.1	SF.2	SF.3		SS.1
SM.1	X				
SM.2	X				
SM.3		X			
SM.4		X			
SM.5		X			
SM.6		X			
SM.7			X		
SM.8			X		
.....					
SM.n					X

Table 2 Mapping of the Security Mechanisms (SM) to the SF and SS

5.2 Self-protection and Non-Bypassability

According to CC part 3, Annex A.1.1, 513 “Self-protection refers to the ability of the TSF to protect itself from manipulation from external entities that may result in changes to the TSF. Without these properties, the TSF might be disabled from performing its security services”.

According to CC part 3, Annex A.1.2, 517 “Non-bypassability is a property that the security functionality of the TSF is always invoked and cannot be circumvented when appropriate for that specific mechanism.

The demonstration that the TOE protects itself from tampering and that the TOE prevents bypass of the SFR-enforcing functionality shall be given in the following description, by analysing different attack methods and applying them to the threats, which are defined in the appropriate Security Target and/or Protection Profile. The analysis assesses the effectiveness of the TOE Security Features and/or the Security Mechanisms (SM) of the TOE to resist against different attack methods. The actual CC supporting document “Application of Attack Potential to Smart Cards“ serves as reference to ensure covering state of the art attacks.

The following description serves as an example of the level of detail that is expected. It is not exhaustive.

5.2.1 Probing

Bus Probing

An obvious point of attack is the bus of the IC. This is used to transfer all the information necessary for executing the program from the peripheral modules such as the memories to the CPU and vice versa. This provides an attacker with the possibility of analyzing program execution and discovering secret data such as keys.

The probing of an IC in principle is a standard failure analysis method in IC design. The TOE comprises a lot of countermeasures to avert such an attack.

This attack alone does not necessarily disclose sensitive information. E.g. in case the bus is encrypted further analysis is needed to gain information.

A commonly used method in the semiconductor industry employs test probes to establish contact with lines and record the information being transferred via a line.

The security mechanism (e.g. SM.x Active Shield) is foreseen to counter that attack method. That security element makes the use of the FIB (bespoke) necessary, as it is not possible to overcome that security element only with probes.

As it is virtually impossible to apply high number of test probes simultaneously and keep them in contact throughout the measurement, it will be necessary to record a program execution in individual portions by repeating the program. For evaluation purposes, these "portions" must be made to coincide in time. This is countered by security elements SM.n and SM.n+1, which ensure that program runs are repeated with different timing.

Additionally the communication over the busses is encrypted by the Security Mechanism SM m.

(Continue description, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack.)

Memory Probing

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for the different memories of the TOE.)

Co-processor Probing

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for the different co-processors of the TOE.)

RNG Probing (Threat/attack against Security Service)

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack.)

<Further attacks related to Physical Probing>

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for the different parts the TOE.)

5.2.2 Manipulation

Memory Manipulation

The purpose of a memory manipulation is to change security relevant data in the TOE. For manipulation bespoke equipment (FIB) is needed. It is not the purpose to randomly change the data. For such a memory manipulation detailed information of the design and high expertise is needed. Even if the location of the relevant memory type is known the Security Mechanism SM.n in combination with the encryption of the stored data implemented by SM.n+m makes it impossible to find the correct memory cell and to manipulate it.

(Continue description, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different description for the different types of memories.)

RNG Manipulation (attack against Security Service)

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack.)

<Other attacks related to Manipulation>

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for the different parts the TOE.)

5.2.3 SPA/DPA attacks

The IC exhibits a power consumption, which is a function of the commands executed and the data used, as both the shunt current in the switching process and the capacitive recharging is dependent on the process taking place at that instant. Various methods of inferring the data being processed from analysis of the power consumption are known (e.g. SPA or DPA). The concerned threat is T.Leak-Inherent.

For this purpose it is first necessary to resolve the power consumption (also by averaging of repeated measurements) to the point that the information becomes visible. This attack consists of the two steps measurement and analysis.

The Security Mechanisms SM.n , SM.n+1, SM.n+m , ... ensure that the data transferred via the bus is encrypted and prevents direct correlation of stored data and power consumption..

(Continue description, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different description for the different types of co-processors.)

5.2.4 Timing attacks

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for the different parts the TOE.)

<Other attacks related to Information Leakage>

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for the different parts the TOE.)

5.2.5 Environmental Stress (V, f, T)

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for different parts of the TOE.)

5.2.6 Light Attacks (Frontside, Backside)

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for different parts of the TOE.)

5.2.7 <Other fault injection attacks>

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for different parts of the TOE.)

5.2.8 Re-activate IC Test mode

(Describe attack or refer to a document where this attack is explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter this attack, if necessary include different descriptions for different parts of the TOE.)

5.2.9 Manipulate/Modify rights for restricted modes of operation

The normal operation mode is the Application Mode (or similar naming) separated from the Test mode. Further modes of operation might exist as sub-modes with different privileges.

(If appropriate describe briefly the relevant modes of operation. The following paragraphs serve as an example.)

Application mode

The application mode refers to the non-test mode. From software point of view it offers two modes which itself could create security domains.

System mode

The system mode will run so-called high privileged tasks.

...

User mode

Software running in user mode has only access to a RAM range specified during system mode. Task running in user mode are further on referenced as low-privileged tasks.

...

Separation of a high- and low-privileged task

The high-privileged task wants to protect the resources it uses from the low-privileged task. Three interfaces are used by the software: The instruction set, the memories and the Special Function Registers.

The instruction set is not restricted, neither in system mode (SMo) nor in user mode (UMo) all instructions are executable. The memory availability can be restricted with the memory segmentation. Since the default state of the MMU in UMo is to deny any access to memory, the SMo task has to create reasonable segments to allow UMo task execution. Covert channels may appear if the SMo task accidentally assigns its own memory to a UMo task or the UMo task gains control over the MMU Segment Table pointer. However, this is in full responsibility of the Smartcard Embedded Software.

The access to the Special Function Registers is dependent on the actual mode. Some SFR are accessible exclusively in one mode, other are shared between different modes. A covert channel may come into existence if the SMo tasks leaves confidential values in SFR that are accessible for the UMo task. The TDS-Documentation gives a detailed listing of all Special Function Registers and the corresponding access rights. A software developer has to consider all Special Function Registers that are available for both SMo and UMo.

(Description has to include how this is organized, do not concentrate on what the software has to do)

Separation between low-privileged tasks

In order to separate different low-privileged tasks from each other, a SMo task has to

- ∅ separate the memory segments of the tasks
- ∅ to clear all Special Function Registers and CPU registers if the execution is switched from one task to another.

The necessary information for the software developer is given in □.

(Add description how this is organized, do not concentrate on what the software has to do)

5.2.10 <Other attacks>

(Describe the different attacks or refer to a document where these attacks are explained, name/describe all the Security Mechanisms and/or Security Features involved and how they are supportive to counter these attack, if necessary include different descriptions for different parts of the TOE.)

6 Bibliography

- Ø Protection Profile BSI-PP-0035-2007
- Ø TDS-Documentation
- Ø User Guidance Manual
- Ø Security Target
- Ø Common Criteria, part 1
- Ø Common Criteria, part 2
- Ø Common Criteria, part 3