Protection Profile for Single Chip Microcontroller equipped with a secure cryptographic unit

Version 1.20

June 15, 2022

NATIONAL INSTITUTE OF
ADVANCED INDUSTRIAL SCIENCE
AND TECHNOLOGY（AIST）

Contents

Revision history

Version	Date	Description
1.20	June 15, 2022	Revised in accordance with comments from evaluation body

Acknowledgements

This Protection Profile (PP) was developed by the SCU Inside System Committee with representatives from universities, national institutes, security service providers, industries, and the Information Technology Security Evaluation Facility. The organizations that directly contributed to the development of this PP include:

National Institute of Advanced Industrial Science and Technology (AIST)

Yokohama National University (YNU)

SECOM Co., Ltd.

Toppan Technical Design Center Co., Ltd.

STMicroelectronics

Electronic Commerce Security Technology Research Association (ECSEC TRA)

ECSEC Laboratory Inc.

0 Preface

0.1 Object of document

This document presents the Protection Profile (PP) to express the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) for a Protection Profile for Single Chip Microcontroller equipped with a secure cryptographic unit (SCU).

0.2 References

[180-4] FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Secure Hash Standard (SHS)

[186-4] FIPS PUB 186-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Digital Signature Standard (DSS)

[202] FIPS PUB 202 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

[800-38A] NIST Special Publication 800-38A Recommendation for Block Cipher Modes of Operation Methods and Techniques

[800-38B] NIST Special Publication 800-38B Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication

[800-38C] NIST Special Publication 800-38C Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality

[800-38D] NIST Special Publication 800-38D Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC

[800-38E] NIST Special Publication 800-38E Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices

[800-38F] NIST Special Publication 800-38F Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping

[800-90B] NIST Special Publication 800-90B Recommendation for the Entropy Sources Used for Random Bit Generation

[800-108] NIST Special Publication 800-108 Recommendation for Key Derivation Using Pseudorandom Functions (Revised)

[800-133] NIST Special Publication 800-133 Revision 1 Recommendation for Cryptographic Key Generation

[1619] IEEE 1619-2018 - IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices

[5639] RFC 5639 Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation

[8032] RFC 8032 Edwards-Curve Digital Signature Algorithm (EdDSA)

[8439] RFC 8439 ChaCha20 and Poly1305 for IETF Protocols

[9797-2] ISO/IEC 9797-2:2011 Information technology - Security techniques - Message Authentication Codes (MACs) - Part 2: Mechanisms using a dedicated hash-function

[10116] ISO/IEC 10116:2017 Information technology - Security techniques - Modes of operation for an n-bit block cipher

[10118-3] ISO/IEC 10118-3:2018 IT Security techniques - Hash-functions - Part 3: Dedicated hash-functions

[14888-3] ISO/IEC 14888-3:2018 IT Security techniques - Digital signatures with appendix - Part 3: Discrete logarithm based mechanisms

[18031] ISO/IEC 18031:2011 Information technology - Security techniques - Random bit generation

[18033-3] ISO/IEC 18033-3:2010 Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers

[19772] ISO/IEC 19772:2009 Information technology - Security techniques - Authenticated encryption

[AAPS] Joint Interpretation Library, Application of Attack Potential for Smartcards and Similar Devices, Version 3.1, June 2020

[AMSS] Joint Interpretation Library, Application of Attack Potential for Smartcards and Similar Devices, Version 2.4, January 2020

[CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, Version 3.1 Revision 5, April 2017.

[CC2] Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components, Version 3.1 Revision 5, April 2017.

[CC3] Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components, Version 3.1 Revision 5, April 2017.

[CEM] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017.

[CPPFDE] collaborative Protection Profile for Full Drive 1 Encryption - Encryption Engine Version 2.0 September 09, 2016

[PP0096] Common Criteria Protection Profile FIDO Universal Second Factor (U2F) Authenticator BSI-PP-CC-0096-V3-2018

[PPTEE] GlobalPlatform Device Committee TEE Protection Profile Version 1.2.1

0.3 Terminology

Terminology	Description
Application software	From the viewpoint of the TOE, application software is user data that uses the encryption service of the SCU via a software gate API.
External entity	Human or IT entity potentially interacting with the TOE from outside its boundary.
Garbage collection	Reclaiming memory occupied by objects that are no longer in use by the program.
Hardware gate	A hardware gate is a hardware component of an access control mechanism for a cryptographic function that accesses a cryptographic engine.

Terminology	Description
Hardware gate key (HGK)	A key as a Root of Trust (RoT).
Intermediate key	A Key Encryption Key (KEK) or Key Wrapping Key (KWK) used to protect a Data Encryption Key (DEK) and MAC key.
Protected storage	Specialised storage for the HGK. It may be an OTP area where the HGK is written by a semiconductor test process or hard coded to be a part of a circuit.
Secrets	In this PP, it refers to data that requires confidentiality protection, such as pre-shared keys, key material, and pre-calculated values.
Submask	A bit string that can be generated and stored in a number of ways.
Software gate	A software gate is a software component of an access control mechanism for a cryptographic function that provides cryptographic function to the application software via a hardware component.
User Keys	Keys used by the application software that are encrypted by the KEK. Data for integrity checks are attached.

0.4 Abbreviation

Abbreviation	Description
AES	Advanced Encryption Standard
API	Application Program Interface, an interface between different parts of a computer program intended to simplify the implementation and maintenance of software
CBC	Cipher Block Chaining
CC	Common Criteria
CCM	Counter with CBC-MAC
CMAC	Cipher-based Message Authentication Code
CPU	Central Processing Unit
CTR	Counter
DEK	Data Encryption Key
DRBG	Deterministic Random Bit Generator
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards-curve Digital Signature Algorithm
GCM	Galois/Counter Mode
FIPS	Federal Information Processing Standard(s)
HGK	Hardware Gate Key
HMAC	Keyed-Hash Message Authentication Code
IEEE	Institute of Electrical and Electronics Engineers
IEC	International Electrotechnical Commission

Abbreviation	Description
IoT	Internet of Things
IP	Intellectual Property, a semiconductor intellectual property
ISO	International Organization for Standardization
IT	Information Technology
IV	Initialisation Vector
KDF	Key Derivation Functions
KEK	Key Encryption Key
KW	Key Wrap
KWK	Key Wrapping Key
KWP	Key Wrap With Padding
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OTP	One Time Programmable
PP	Protection Profile
RBG	Random Bit Generator
RoT	Root of Trust
SAR	Security Assurance Requirement
SCU	Secure Cryptographic Unit
SFP	Security Functional Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SoC	System on a Chip
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Function
XTS	XEX (XOR – Encrypt – XOR) based tweaked-codebook mode with cipher text stealing

1 PP introduction

1.1 PP reference identification

PP Reference	Protection Profile for Single Chip Microcontroller equipped with a secure cryptographic unit
Version	1.20
Date	June 15, 2022

1.2 TOE overview

1.2.1 Composition of the TOE

The Target of Evaluation (TOE) is a Single Chip Microcontroller equipped with a secure cryptographic unit (SCU). The SCU consists of a cryptographic engine, and software and hardware gates that can access the cryptographic engine via “software gate APIs.” The assumed TOE is a built-in memory type in a single-chip microcontroller. An external memory type that has large-scale memory outside the single-chip microcontroller is assumed as another type of TOE; however, this is a future challenge.

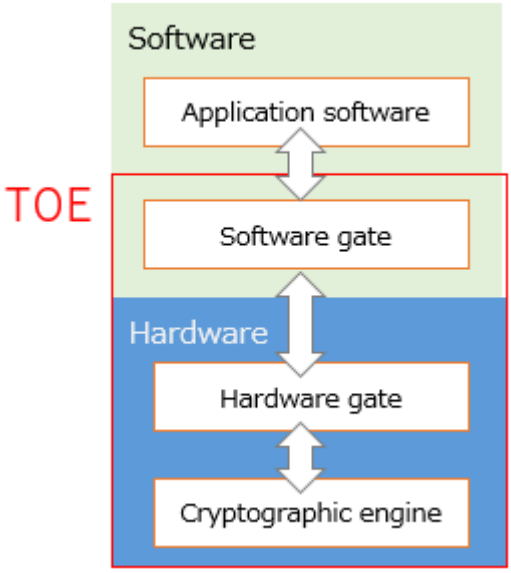


Figure 1-1 Conceptual diagram of the SCU

The TOE is generally distributed in the form of a SoC. In the case of the TOE with built-in memory, it is packaged by mounting it on a single die. This SoC is soldered on a board on which various circuits necessary for embedded device applications are mounted, and the board is placed in the housing of the embedded device.

Figure 1-2 shows an example of the TOE configuration. The blue line shows the physical boundary of the TOE and the configuration of a typical microcontroller equipped with an SCU. The red line shows the logical boundary of the TOE. In Figure 1-2, the application software is logically located outside the TOE and uses cryptographic functions through the software and hardware gates. The application software is stored in non-volatile memory inside the physical boundary of the TOE.

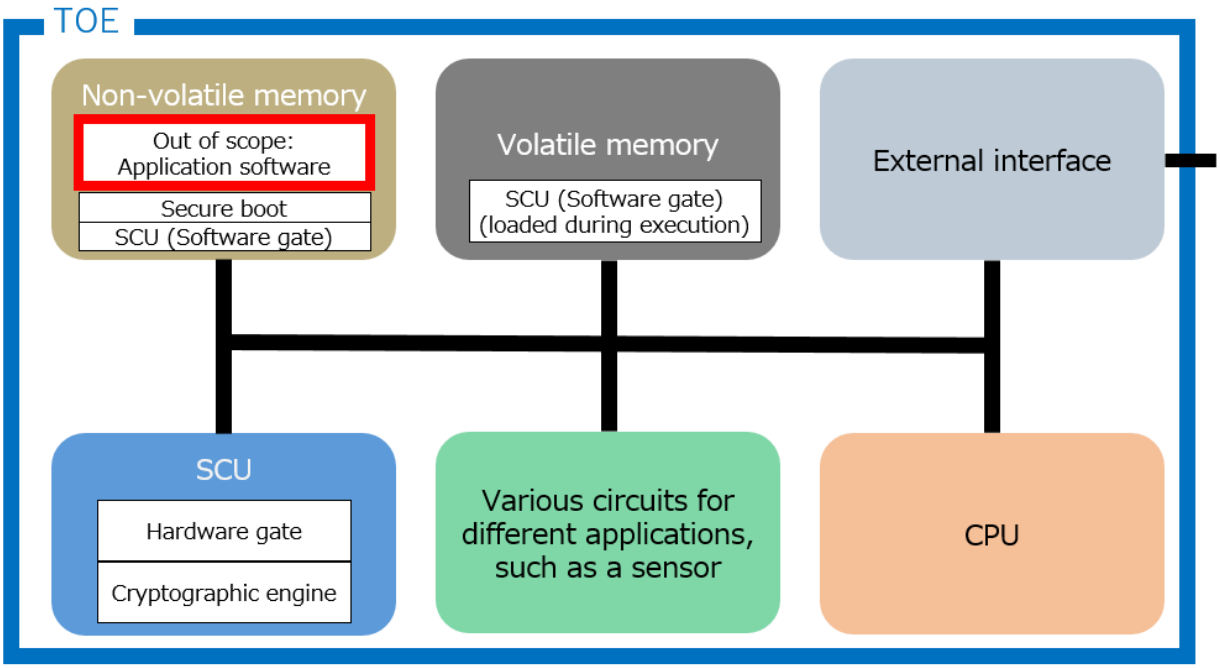


Figure 1-2 TOE configuration example

The software and hardware gates are mechanisms for controlling access to the cryptographic functions, and they correspond to software and hardware components of the access control function. Access to the cryptographic functions by the application software is legitimate and should be permitted, but access by others must be denied. The TOE access control mechanism is implemented to distinguish between these accesses.

The software and hardware gates operate as follows. Each time the hardware gate receives cryptographic command data directed at it, it transitions its internal state. This is uniquely determined by the previous internal state and the current input (cryptographic command data). If the hardware gate knows the correct transition data of the internal state, it can compare the result of the transition by the current input with the correct transition data to determine whether the input command data is legitimate.

The software gates provide information to manage the state transitions of the hardware gates. All patterns of access to the cryptographic functions, which are unique and predicted in advance by the developer, are known only to the developer. The internal state transitions of the hardware gate associated with these access patterns are calculated in advance and stored in the software gate. When the software gate receives the access command data to the cryptographic function from the application software, it transfers both the access command data and the next internal state transition data of the hardware gate to the hardware gate. The hardware gate checks the internal state transition data passed to it against the internal state transition results from the command data received at the same time, and if the two match, it determines that the received command data is legitimate. Software gates, which contain internal state transition data to be passed to hardware gates, are stored in non-volatile memory in the TOE at the time of TOE manufacturing and cannot be generated except by developers. In other words, the access pattern to the cryptographic function calculated in advance by the developer will be executed, but if an emulator or other device tries to use the cryptographic function with any other pattern, it will not be able to provide the encrypted internal state transition data and the hardware gate will deny access.

The components labeled as software and hardware gates in Figure 1-2 depend on the implementation of the SCU; to protect the integrity of the contents of the memory located outside the SCU and the integrity of the cipher processing, the application software should be developed to use software gates appropriately.

The SCU has a self-protection function to protect its own security function. The TOE stores the HGK, which is the RoT, and information that uniquely identifies the TOE to external entities in the TOE. It is the responsibility of the TOE developer to generate a random number HGK with sufficient entropy.

The TOE's secure boot program extracts the software gate and application software into RAM at startup, verifies the integrity of the software gate, and verifies the integrity (and optionally the authenticity) of the application software. The TOE also verifies the integrity and authenticity of the application software when updating it, and updates it after successful verification. Here, authenticity refers to the property that the application software was developed by a legitimate application software developer.

1.2.2 Security features of the TOE

The SCU provides a set of cryptographic functions as a basis for security. The TOE provides the functions to application software via software gate. The application software implements security functions such as communication protocol, memory encryption, and identification/authentication using cryptographic functions. The TOE also implements the cryptographic functions and self-protection functions that protect the security functions using the cryptographic function.

The main security functions provided by the TOE are as follows. These functions are baseline requirements, hence TOE’s mandatory requirements.

- Monitoring access to the cryptographic function: The ability to detect and respond to unauthorized use of the cryptographic function by an attacker through the cooperative operation of the software and hardware gates.
- Self-protection function: The ability to prevent unauthorized use of leakage during SCU operation and the ability to detect and respond to physical attacks.
- Secure boot function: The ability to verify the integrity of the software gate and the application software during startup.
- Store keys: The ability to provide key storage whose confidentiality and integrity are protected by cryptography in order to store the keys in the memory area of the TOE outside the SCU.
- Import user keys: The ability to import key storage containing user keys and secret information from external entities to the TOE while protecting confidentiality.
- Update function: The ability to update after verifying the authenticity and integrity of the application software.

The cryptographic functions for achieving the baseline functions of the TOE are as follows. The Security Target (ST) author selects the required Security Functional Requirement (SFR) from Chapter 7.

- Encryption/Decryption: To protect confidentiality, a plaintext is encrypted into a cipher text and a cipher text is decrypted into a plaintext.
- Digital signature verification: Verifying the digital signature for authenticity and integrity verification.
- Calculation of hash value: cryptographic hash functions calculate hash values.
- MAC generation and verification: Attaching a MAC and verifying the integrity of data with the MAC.
- Random bit generation: The TOE generates random bits and provides them for the application software.
- Using salt, nonce and generating IV: Appropriate use of salts and nonces required for cryptographic functions and generating IVs.
- Deriving keys: It derives keys.

- Encrypt key: It encrypts the keys using KEKs.

The cryptographic functions provided by the TOE to the application software via the software gate are as follows. These functions are optional features, and the ST author selects the necessary SFRs from Chapter 8.

- Key generation: Keys suitable for the cryptographic algorithm and use are generated by the Random Bit Generator (RBG) of the TOE.
- Destruction of keys and key materials: Making the keys and key materials stored in the volatile memory unrecoverable. Note that the keys stored in the key storage of non-volatile memory are encrypted and are not expected to be destroyed.
- Digital signature generation: Generate a digital signature for the protection of authenticity and integrity.

1.2.3 Use case of the TOE

The TOE equipped with the SCU is a microcontroller for embedded devices known as IoT edge devices such as sensors, actuators, and surveillance cameras. It is conceivable that the microcontroller, which is the TOE, has functions to process raw data collected by those devices, securely store the processed information, and securely transfer it outside the device. The SCU is a RoT for secure processing. The embedded device manufacturer defines a security policy of the embedded device, decides the information for protecting confidentiality and that for protecting integrity, and implements the application software.

The TOE protects the confidentiality, integrity, and authenticity of data handled by the application software by using its services in accordance with the instructions of the application software. That is, the TOE cannot determine which data should be protected. For example, the TOE cannot protect the confidentiality of data stored in memory in plain form or data sent to the outside in plain form. Therefore, developers who implement the application software using the TOE must identify the data they want to protect and must protect the data using the SCU.

1.2.4 Roles

The main target audience for this PP is TOE developers. TOE developers purchase SCU IPs or develop them themselves, develop and manufacture single chip microcontrollers by integrating necessary components such as CPU and memory. The TOE developers also store an HGK in the SCU when the TOE is manufactured. TOE manufacturing may be manufactured at the TOE developers' factory or outsourced to a manufacturing company.

TOE users are embedded device developers who purchase the TOE and incorporate the application software into it to develop and manufacture embedded devices. Therefore, TOE guidance will be distributed to embedded device developers.

The TOE developers never manage embedded devices and do not need to inform the embedded device developer about the HGK. It is assumed that the user keys will be imported to the embedded device developer before TOE delivery. Optionally, to update the user keys, the embedded device developer can also implement a software gate API call in the application software to import the user keys.

End consumers purchase and use the embedded device equipped with the TOE. Management of the embedded device may occur via the application software outside the TOE. However, because the TOE is based on the application software implementation, management is related to the embedded device, not the TOE.

1.2.5 Life cycle of the TOE

The TOE developer shall secure the process from TOE development in Phase 1 to TOE manufacturing in Phase 4, and the delivery of the TOE to embedded device developers in Phase 6. The TOE developer or key installation providers must also secure the importing user key in Phase 5.

The environment for application software development in Phase 3, embedded device manufacturing in Phase 6, and distribution to end consumers in Phase 7 are outside the scope of this PP, but it is assumed that the embedded device developer who purchased the TOE will take responsibility for maintaining the development environment security.

Phase 1: Developing hardware

Development of the TOE. The TOE developer purchases an IP of the SCU or develops the SCU and constructs the hardware TOE together with components such as a CPU.

Phase 2: Purchasing software or developing program

The TOE developer purchases the software gate for secure use of the cryptographic engine from the SCU IP vendor or develops their own software gate. The TOE developer purchases a secure boot program from the SCU IP vendor or develops their own secure boot program.

Phase 3: Developing the application software

The embedded device developer develops the application software for embedded devices. When requesting the TOE manufacturer to install the application software, the embedded device developer sends the application software to the TOE developer. If the embedded device developer installs the application software, it will be installed in Phase 6.

Phase 4: Manufacturing the TOE

The TOE developer manufactures the TOE, writes the HGK, and installs the secure boot and software gate into the non-volatile memory of the TOE. If requested by the embedded device developer, the TOE developer receives the application software developed by the embedded device developer in Phase 3 and loads it into the non-volatile memory of the TOE. Note that the application software may be mounted on the TOE in Phase 6 instead of this phase. The manufactured TOE becomes a product after a developer test.

Phase 5: Importing the user keys

The TOE developer generates a data object (key storage) to store user keys and secret information. The TOE developer receives the user key and secret information used by the application software from the embedded device developer and stores them in key storage, and the entire key storage is encrypted and assigned a MAC. The TOE writes the key storage to the TOE's non-volatile memory via a software gate. Key importing may be performed by the TOE developer or outsourced to a key installation provider. In any case, it is necessary that key delivery and importing are performed in a secure environment. After writing the key, the TOE will be distributed from Phase 5 to Phase 6.

Phase 6: Manufacturing the embedded device

The embedded device developer manufactures the embedded device and mounts the TOE on the embedded device. In a number of cases, the embedded device developer installs the application software into the TOE. This development process may be divided into the development of a board on which the TOE is mounted and that of the embedded device on which it is mounted. The TOE in this process is assumed to be handled securely. The completed embedded device is distributed to the end consumer.

Phase 7: Operating by end consumers

The final phase of the TOE life cycle. It is used under the assumed operating environment with the TOE installed in the embedded device. The threats assumed by this PP occur during this operation phase.

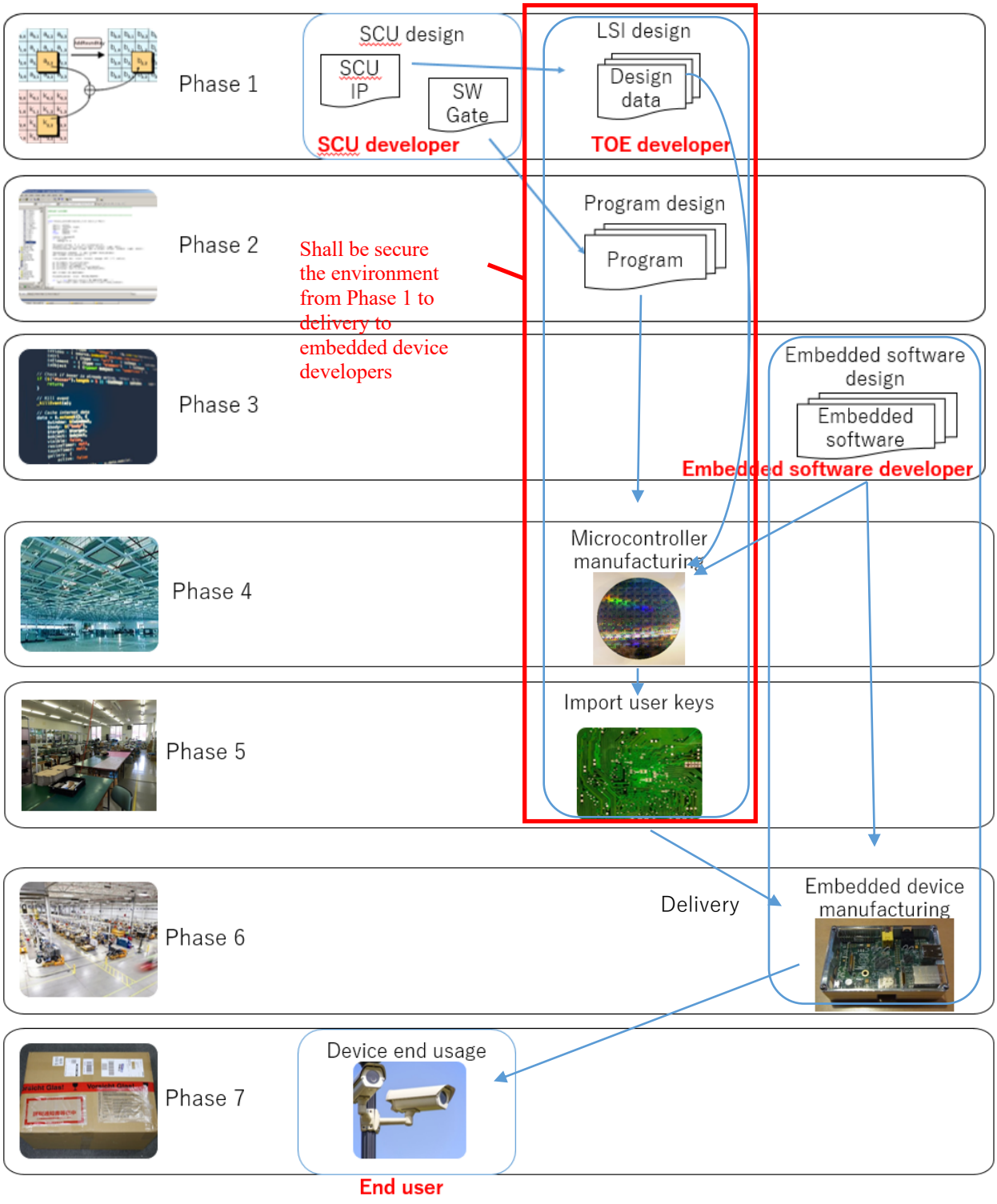


Figure 1-3 Life cycle of the TOE

1.2.6 Protection of keys

The TOE provides key storage services and keeps cryptographic keys used by the cryptographic engine confidential and secure by encrypting them. The integrity of a key is protected by a MAC. As an example, when the cryptographic key used in the hardware gate is held in non-volatile memory outside the SCU and inside the TOE, the following processing is performed. First, at the TOE developer's factory or outsourced key installation provider in Phase 5, the KEK and MAC keys are derived from the HGK written inside the SCU. Next, the key storage (which contains user keys and other data) used in the TOE is encrypted using KEK, and a MAC is assigned to the encrypted key storage using the MAC key. In this way, the key storage is stored into the memory in the TOE but out of the SCU in a state of being encrypted and attached with a MAC.

The TOE has a data object (protected storage) created by the manufacturer at the time of manufacturing, and the TOE developer stores the HGK in that data object. For example, the TOE developer generates an HGK with an RBG outside the TOE and embeds it in the TOE during the manufacturing process. The HGK guarantees the integrity of the TOE and is the starting point for permissions to other data objects (key storage). Since the HGK must be protected, the TOE needs a self-protection function.

2 Conformance claims

This PP is CC version 3.1 revision 5 (Japanese) conformant.

This PP conforms to CC part 2 extended and CC part 3 extended. Extended components are defined in Chapter 6.

This PP does not claim conformance to any another PP.

The TOE conformance to this PP is EAL 1 augmented with ASE_SPD.1, ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, ALC_FLR.1, AVA_VAN.2, and AVA_SCU_EXT.1. AVA_SCU_EXT.1 is defined in Chapter 6.

This PP requires strict conformance of the ST or PP claiming conformance to this PP.

This PP satisfies the following Assurance Families: APE_CCL.1, APE_ECD.1, APE_INT.1, APE_OBJ.2, APE_REQ.2 and APE_SPD.1.

3 Security problem definition

3.1 Assets

The TOE protects following assets ‘As.’

As.SCU	Integrity of security services provided for the application software.
As.ConfUD	Confidentiality of user data that requires confidentiality.
As.IntegUD	Integrity of user data that requires integrity.

The TOE protects confidentiality of TSF data by the self-protection function of the hardware. The TOE protects user data that exists in non-volatile memory by the cryptographic function of the TOE.

User data may include user keys (a private key for digital signature generation of data, a private key for device authentication, a public key for verification of the application software a public key for certificate verification data, and so on), certificate, data exchanged via the software gate API, and the application software. The TOE cannot determine the necessary confidentiality and integrity of user data. Therefore, the user must securely store the user data that requires confidentiality and integrity by using the TOE function.

TSF data are an HGK, a key for decrypting user keys in the key storage, data for verification of the integrity of the key storage, data for verification of the integrity of the software gate, an IV, internal state transition data, and a chip ID.

3.2 Threats

T.Internal_Access

An attacker may attempt to tamper with the application software or the software gate and then use the TOE's cryptographic functions without permission to disclose or modify the user data.

Rationale: Following SFRs prevent modified application software or modified software gate from accessing the hardware inside the TOE to use the TOE’s cryptographic functions.

- FDP_IFC.1/API, FDP_IFF.1/API specifies requirements for operating cryptographic functions by the application software that is an external entity. Only when the state transition of data of the software gate is verified, data is output from the cryptographic function to the external entity. That is, only the correct use of the cryptographic function is accepted.
- FDP_MFW_EXT.1 is called by FPT_TST.1 to verify the integrity and, if needed, authenticity of the application software at startup.
- FPT_TST.1 defines the verification of the integrity of the software gate at startup and supports the validation of the state transition data of the software gate by FDP_IFC.1/API and FDP_IFF.1/API.
- FPT_FLS.1/SG maintains a secure state even if the integrity of the state transition of data of the software gate is compromised.
- FPT_FLS.1/SB maintains a secure state even if the integrity of the software gate and application software and, if needed, authenticity of the application software is compromised at startup.

T.Weak_Import

An attacker may abuse the key storage import function to disclose or tamper with user data.

Rationale: According to the following SFRs, importing of the key storage with encrypted integrity verification data attached and integrity verification are performed to prevent the import of unauthorized user data.

- FDP_IFC.1/Import, FDP_IFF.1/Import specifies the requirements for operating the import functions by the application software that is an external entity. Only when the state transition of data of the software gate is verified can the user data be stored in the key storage via the import function. That is, only the correct use of the import function is accepted.
- FDP_UIT.1 verifies the integrity of user data to be imported into key storage.

T.Unauthorized_Update

An attacker may install unauthorized application software on the TOE to expose user data of the embedded device or disrupt the device's services. Alternatively, an attacker may illegally roll back to a version with a security failure to disclose the user data of the embedded device or disrupt embedded device services.

Rationale: According to the following SFRs, the TSF obtains the correct version of the application software, updates the application software, and verifies the updated application software.

- FPT_TUD_EXT.1 queries the application software for its current version, triggers the update, and verifies the updated application software before installation.
- FPT_RPL.1 prevents rollback attempts.
- FPT_FLS.1/UD preserves a secure state when an integrity or authenticity error of the application software occurs.

T.Weak_Crypto

An attacker may disclose or alter the user data by exploiting improperly selected encryption algorithms, key generation method, key lengths, key destruction method, or an RBG.

Rationale: The TOE counters this threat by implementing an RBG with a sufficient entropy source and cryptographic algorithms with sufficient key length on the basis of an approved standard, and providing it to the application software, as defined in the following SFRs.

- (Option) FCS_CKM.1/AK generates asymmetric keys.
- (Option) FCS_CKM.1/SK generates symmetric keys.
- (Option) FCS_CKM.4 ensures that keys and key materials in the volatile memory are destroyed in such a way as to prevent future recovery.
- (Selection) FCS_COP.1/SKC encrypts and decrypts using symmetric key algorithms.
- (Selection) FCS_COP.1/KeyEnc performs key encryption and decryption.
- (Selection) FCS_COP.1/Hash uses hashing mechanisms.
- (Selection) FCS_COP.1/MAC calculates MAC.
- (Option) FCS_COP.1/SigGen generates digital signatures.
- (Selection) FCS_COP.1/SigVer verifies digital signatures.
- (Selection) FCS_KDF_EXT.1 performs key derivation.
- (Selection) FCS_RBG_EXT.1 performs random bit generation.

- (Selection) FCS_SNI_EXT.1 ensures that the salt, nonce, and initialisation vector used by the TOE do not adversely affect key strength.

T.Leak_Inherent

An attacker may expose the user data or TSF data like SCU's cryptographic keys by observing and analyzing changes in the TOE's power consumption during cryptographic operations.

Rationale: It mitigates leakage of unnecessary information to radiated electromagnetic waves and power consumption when SCU processes user and TSF data, making it difficult for attackers to expose useful data by performing statistical processing.

- FPT_EMS_EXT.1 mitigates the leakage of user data and TSF data from the TOE.

T.Phys_Probing

By physically probing the inside of the TOE, an attacker may expose or modify the user data of the TOE like cryptographic keys, or other TSF data that is useful for other attacks.

Rationale: The following SFRs are to counter this threat of modifying or acquiring the user data by using equipment used for semiconductor analysis to photograph a memory cell or physically contact the inside of the TOE.

- FPT_PHP.3 counters physical probing.
- FCS_STG_EXT.1 implements key storage outside the SCU.
- FCS_STG_EXT.2 uses cryptography to ensure confidentiality of the key storage outside the SCU.
- FCS_STG_EXT.3 uses cryptography to ensure integrity of the key storage outside the SCU.

T.Phys_Manipulation

An attacker may modify the user data and encryption keys stored in the cryptographic function by physically manipulating the inside of the TOE or modify the security mechanism of the TOE for other attacks.

Rationale: The following SFRs are aimed at countering this threat that directly alters information assets or uses them as a stepping stone for other attacks by conducting physical operations inside the TOE.

- FPT_PHP.3 counters physical tampering.
- FCS_STG_EXT.1 maintains key storage outside the SCU.
- FCS_STG_EXT.2 uses cryptography to ensure confidentiality of the key storage outside the SCU.
- FCS_STG_EXT.3 uses cryptography to ensure integrity of the key storage outside the SCU.

3.3 Organizational security policy

There is no organizational security policy.

3.4 Assumptions

A.Trusted_User

The embedded device developer appropriately protects data stored outside the TOE.

The security objective for the operational environment is OE.Trusted_User.

4 Security objectives

This chapter describes only security objectives for the operational environment in accordance with ASE_OBJ.1.

4.1 Security objectives for the operational environment

OE.Trusted_User

The embedded device developer follows the guidance of the TOE regarding the protection of data held outside the TOE.

Rationale: This assumption is fulfilled if the embedded device developer implements protection of data held outside the TOE in accordance with the guidance of the TOE.

5 Security requirements

The conventions used in the SFR descriptions are as follows:

(1) Unaltered SFRs are stated in the form used in [CC2] or their Extended Component Definition (ECD).

(2) Refinement made in the PP: the added/removed text is indicated with **bold text**/~~strikethroughs~~. When text is substituted, the description will result in an additional description in bold.

(3) Selections:

a) Wholly or partially completed in the PP: the selection values (i.e., the selection values adopted in the PP or the remaining ones available for the ST) are indicated with underlined text.

E.g., ECD [selection: physical, non-physical true, deterministic, hybrid physical, hybrid deterministic] might become [physical] (completion) or [selection: physical, non-physical true] (partial completion) in the PP.

b) A number of SFRs include selections that determine or constrain other assignments or selections. In these cases, a table follows the requirement in which each row of the table defines a permitted set of choices. Each row includes a unique identifier defined solely to provide a label for the selection set.

(4) Assignment wholly or partially completed in the PP: indicated with *italicized text*.

(5) Assignment completed within a selection in the PP: the completed assignment text is indicated with *italicized and underlined text.*

E.g., "[selection: change_default, query, modify, delete, [*assignment: other operations*]]" in [CC2] or an ECD might become "[change_default, [*select_tag*]]" (completion of both selection and assignment) in the PP;

(6) Iteration: indicated by adding a string starting with "/" (e.g., "FCS_COP.1/Hash").

(7) Extended SFRs and an SAR are identified by having a label '_EXT' at the end of the name.

5.1 Security functional requirements

5.1.1 Cryptographic support

FCS_STG_EXT.1 Secure key storage

Hierarchical to: No other components

Dependencies: No dependencies

FCS_STG_EXT.1.1 The TSF shall provide [*software-based*] secure key storage for asymmetric private keys and [selection: *symmetric keys, secrets, no other keys*].

FCS_STG_EXT.1.2 The TSF shall support the capability of importing keys/secrets into the TOE upon request of [*the application software*]**.**

FCS_STG_EXT.1.3 The TSF shall be capable of destroying keys/secrets in the secure key storage upon request of [*the application software*]**.**

Application Note 1

The secure key storage is implemented in software in the memory inside the TOE, so this requirement covers it.

FCS_STG_EXT.2 Key storage encryption

Hierarchical to: No other components

Dependencies: FCS_COP.1

FCS_STG_EXT.2.1 The TSF shall protect the confidentiality of [*all software-based key storage*] using the following method: [assignment: *as specified in FCS_COP.1/KeyEnc*].

Application Note 2

The ST author selects an SFR from Chapter 7 t7o decrypt the key storage. When re-encrypting, the ST author selects SFRs to encrypt and decrypt from Chapter 7.

FCS_STG_EXT.3 Key integrity protection

Hierarchical to: No other components

Dependencies: FCS_COP.1

FCS_STG_EXT.3.1 The TSF shall protect the integrity of [*all software-based key storage*] by using [selection:

- *A hash of the stored key in accordance with FCS_COP.1/Hash [selection: SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512];*
- *A MAC of the stored key in accordance with FCS_COP.1/MAC [selection: HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, CMAC-AES-128, CMAC-AES-256];*
- *Symmetric key encryption in accordance with FCS_COP.1/SKC [selection: AES_CCM, AES_GCM, AES_KWP, AES_KW];*
- *A digital signature of the stored key in accordance with FCS_COP.1/SigVer using an asymmetric key that is protected in accordance with FCS_STG_EXT.2*

]

Application Note 3

The ST author selects an SFR from Chapter 7 to protect the integrity of the key storage.

5.1.2 User data protection

FDP_IFC.1/API Subset information flow control (API)

Hierarchical to: No other components.

Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1/API The TSF shall enforce the [*embedded device information flow control SFP*] on [*subject: cryptographic functions of the TOE, information: API call for the subject and API response from the subject, and operations: input to the API, execution of cryptographic functions and output from the API response*].

FDP_IFC.1/Import Subset information flow control (Import)

Hierarchical to: No other components.

Dependencies: FDP_IFF.1 Simple security attributes

FDP_IFC.1.1/Import The TSF shall enforce the [*import information flow control SFP*] on [*subject: non-volatile memory of the TOE, information: the key storage, and operations: import*].

FDP_IFF.1/API Simple security attributes (API)

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control, FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/API The TSF shall enforce the [*embedded device information flow control SFP*] on the basis of the following types of subject and information security attributes: [*subject: cryptographic functions of the TOE, information: software gate API call for the subject and software gate API response from the subject, the security attributes of the subject: Internal state transition data to verify the integrity of pre-built software gate API calls, the security attribute of the information: Internal state transition results provided to the function that verifies the integrity of software gate API calls*].

FDP_IFF.1.2/API The TSF shall permit an information flow between a controlled subject and information via a controlled operation if the following rules hold: [*permits the information flow of software gate API calls to the subject and software gate API responses from the subject, that is, the output of cryptographic operations, only when the internal state transition data and result calculated by TSF match and the integrity verification of the software gate API call is successful*].

FDP_IFF.1.4/API The TSF shall explicitly authorize an information flow on the basis of the following rules: [*No rules, based on security attributes that explicitly authorize information flows*].

FDP_IFF.1.5/API The TSF shall explicitly deny an information flow on the basis of the following rules: [*No rules, based on security attributes that explicitly deny information flows*].

Application Note 4

The information security attribute is a value calculate for each software gate API call, and the TSF compares the value with reference data to verify authenticity. If a cryptographic algorithm is used for the method of computing the internal state transitions to verify integrity, the implemented algorithm should be selected from Chapter 7.

If the pre-built internal state transition data is to be protected by cryptography, the implemented algorithm should be selected from Chapter 7.

FDP_IFF.1/Import Simple security attributes (Import)

Hierarchical to: No other components.

Dependencies: FDP_IFC.1 Subset information flow control, FMT_MSA.3 Static attribute initialisation

FDP_IFF.1.1/Import The TSF shall enforce the [*import information flow control SFP*] on the basis of the following types of subject and information security attributes: [*subject: non-volatile memory of the TOE, information: the key storage, the security attributes of the subject: none, the security attribute of the information: value used to verify the integrity of the key storage.*].

FDP_IFF.1.2/Import The TSF shall permit an information flow between a controlled subject and information via a controlled operation if the following rules hold: [

enables the import of the key storage into the subject only when the value used to verify the integrity of the key storage matches that calculated by the TSF to verify the integrity of the key storage.].

FDP_IFF.1.4/Import The TSF shall explicitly authorize an information flow on the basis of the following rules: [*No rules, based on security attributes that explicitly authorize information flows*].

FDP_IFF.1.5/Import The TSF shall explicitly deny an information flow on the basis of the following rules: [*No rules, based on security attributes that explicitly deny information flows*].

Application Note 5

The ST author selects an SFR from Chapter 7 to verify the integrity. The selections are hashes in accordance with FCS_COP.1/Hash, a MAC in accordance with FCS_COP.1/MAC, and authenticated encryption in accordance with FCS_COP.1/SKC.

FDP_MFW_EXT.1 Basic software integrity and authenticity (Secure boot)

Hierarchical to: No other components

Dependencies: FCS_COP.1

FDP_MFW_EXT.1.1 The TSF shall have the ability to verify [the integrity] of **the application software**.

FDP_MFW_EXT.1.2 The TSF shall provide a capability to generate evidence of [the integrity] of **the application software**.

Application Note 6

The TOE ensures the integrity of the application software (Secure boot). The ST author selects an SFR from Chapter 7 to verify the integrity. The selections are hashes in accordance with FCS_COP.1/Hash, a MAC in accordance with FCS_COP.1/MAC, and authenticated encryption in accordance with FCS_COP.1/SKC.

FDP_UIT.1 Data exchange integrity

Hierarchical to: No other components.

Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] [FTP_ITC.1 Inter-TSF trusted channel, or FTP_TRP.1 Trusted path]

FDP_UIT.1.1 The TSF shall enforce the [*import information flow control SFP*] to [*receive*] user data in a manner protected from [*modification*] errors.

FDP_UIT.1.2 The TSF shall determine on receipt of user data whether [*modification*] has occurred.

Application Note 7

The TOE ensures the integrity of the application software (Secure boot). The ST author selects an SFR from Chapter 7 to verify the integrity. The selections are hashes in accordance with FCS_COP.1/Hash, a MAC in accordance with FCS_COP.1/MAC, and authenticated encryption in accordance with FCS_COP.1/SKC.

5.1.3 Protection of the TSF

FPT_EMS_EXT.1 Mitigation of leak from the TOE

Hierarchical to: No other components

Dependencies: No dependencies

FPT_EMS_EXT.1 The TOE shall not emit [*power consumption, electromagnetic emanation*] over its [*power line of the TOE, surface of the TOE*] in such an amount that these emissions enable access to [*list of type of TSF data as follows*] and [*list of type of user data as follows*].

Table 5-1 Data to protect against side-channel attacks

list of type of TSF data	list of type of user data
An encryption key of the key storage [*assignment: other TSF data]*	*assignment: list of type of user data*

FPT_FLS.1/SB Failure with preservation of secure state (Secure boot)

Hierarchical to: No other components

Dependencies: No dependencies

FPT_FLS.1.1/SB The TSF shall preserve a secure state when the following types of failures occur: [*software gate integrity violation at startup, application software integrity violation at startup*, **[selection: application authenticity violation at startup, [assignment: other violation]]]**.

Application Note 8

This SFR maintains a secure state when failure of the verification of the software gate and application software by FPT_TST.1 occurs.

FPT_FLS.1/SG Failure with preservation of secure state (Software gate)

Hierarchical to: No other components

Dependencies: No dependencies

FPT_FLS.1.1/SG The TSF shall preserve a secure state when the following types of failures occur: [*failure of internal transition data integrity verification as specified in FPT_TST.1*].

Application Note 9

This SFR covers the case where a mismatch occurs between the internal state transition data initiated by FPT_TST.1 and the calculated one.

FPT_FLS.1/UD Failure with Preservation of Secure State (Trusted update)

Hierarchical to: No other components

Dependencies: No dependencies

FPT_FLS.1.1/UD The TSF shall preserve a secure state when the following types of failures occur: *[integrity or authenticity error of the application software for update*].

Application Note 10

This requirement is intended to verify the integrity and authenticity of the application software being updated.

FPT_PHP.3 Resistance to physical attack

Hierarchical to: No other components

Dependencies: No dependencies

FPT_PHP.3.1 The TSF shall resist [*physical manipulation and probing*] to the [*TSF*] by responding automatically such that the SFRs are always enforced.

Refinement: The TSF will implement appropriate mechanisms to continuously counter physical manipulation and probing. Due to the nature of these attacks (especially manipulation) the TSF can by no means detect attacks on all of its elements. Therefore, permanent protection against these attacks is required to ensure that SFRs are enforced. Hence, "automatic response" indicates (i) the assumption that there might be an attack at any time and (ii) countermeasures are provided at any time.

Application Note 11

The ST shall describe the automatic response of the TOE. The SFRs are enforced if the TOE stops operation or does not operate at all if a physical manipulation or probing attack is detected and the security cannot be ensured in another way.

FPT_RPL.1 Replay detection (rollback)

Hierarchical to: No other components

Dependencies: No dependencies

FPT_RPL.1.1 The TSF shall detect replay for the following entities: [*application software of a previous version*].

FPT_RPL.1.2 The TSF shall [*prevent the execution of the loaded application software and perform* **[**selection, choose one of: [*assignment: other actions*], no other actions**]]** when replay is detected.

Application Note 12

When loading application software is requested, the TSF ensures that the verified application software version is equal to or higher than the previously verified version. Loading an older application software version could expose known vulnerabilities.

FPT_TST.1 TSF testing

Hierarchical to: No other components

Dependencies: No dependencies

FPT_TST.1.1 The TSF shall run a suite of self-tests ***during initial startup,*** [*selection: periodically during normal operation, at the request of* ***the application software****, at the conditions [assignment: conditions under which self-test should occur]*] to demonstrate the correct operation of [*the cryptographic functions*].

FPT_TST.1.2 The TSF shall provide **the hardware gate** with the capability to verify the integrity of [*the internal state transition data*].

FPT_TST.1.3 The TSF shall provide **the self-test during initial startup** with the capability to verify the integrity of [*the software gate*].

Application Note 13

The purpose of this requirement is to verify the completeness of the state transitions by matching the internal state transition results with the internal state transition data, so that the verification of the software gate itself can be performed. Self-testing during initial commissioning verifies the integrity of the software gate, the integrity of the application software, and optionally the authenticity (Secure boot). The TOE ensures the integrity of the application software (Secure boot). The ST author selects an SFR from Chapter 7 to verify the software gate integrity. The

selections are stored hashes in accordance with FCS_COP.1/Hash, a MAC in accordance with FCS_COP.1/MAC, authenticated encryption in accordance with FCS_COP.1/SKC, and signature verification in accordance with FCS_COP.1/SigVer.

FPT_TUD_EXT.1 Trusted update

Hierarchical to: No other components

Dependencies: FCS_COP.1

FPT_TUD_EXT.1.1 The TSF shall provide [*the software gate*] the ability to query the current version of **the application software**.

FPT_TUD_EXT.1.2 The TSF shall provide [*the software gate*] the ability to initiate updates to **the application software**.

FPT_TUD_EXT.1.3 The TSF shall verify updates to **the application software** using a [selection: *digital signature in accordance with FCS_COP.1/SigVer, a MAC in accordance with FCS_COP.1/MAC*] prior to installing those updates.

Application Note 14

This requirement provides the ability for the TOE to update the application software. The TOE shall ensure the integrity and authenticity of the application software by verifying the application software to be updated upon update. The ST author selects the algorithm to be implemented from Chapter 7.

5.2 Security assurance requirements

The SARs defined in this PP are EAL 1 + ASE_SPD.1 + ADV_ARC.1 + ADV_FSP.2 + ADV_TDS.1 + ALC_FLR.1 + AVA_VAN.2 + AVA_SCU_EXT.1.

5.3 Security requirements rationale

5.3.1 Security functional requirements rationale

Table 5-2 shows the dependencies satisfied in the SFRs of baseline requirements.

Application Note 15

If selection-based SFRs are selected, the ST author shall indicate the dependencies that will be met and describe the rationale for any dependencies that will not be met.

If option-based SFRs are added, the ST author shall indicate the dependencies that will be met and describe the rationale for any dependencies that will not be met.

Table 5-2 Baseline security functional requirements

Requirements	CC dependencies	Satisfied dependencies
FCS_STG_EXT.1	No dependency	NA
FCS_STG_EXT.2	FCS_COP.1	Note: The ST author selects SFR.
FCS_STG_EXT.3	FCS_COP.1	Note: The ST author selects SFR.
FDP_IFC.1/API	FDP_IFF.1	FDP_IFF.1/API
FDP_IFC.1/Import	FDP_IFF.1	FDP_IFF.1/Import

Requirements	CC dependencies	Satisfied dependencies
FDP_IFF.1/API	FDP_IFC.1 and FMT_MSA.3	FDP_IFC.1/API Note: FMT_MSA.3 is not satisfied.
FDP_IFF.1/Import	FDP_IFC.1 and FMT_MSA.3	FDP_IFC.1/Import Note: FMT_MSA.3 is not satisfied.
FDP_MFW_EXT.1	FCS_COP.1	Note: The ST author selects SFR.
FDP_UIT.1	[FDP_ACC.1 or FDP_IFC.1] and [FTP_ITC.1 or FTP_TRP.1]	FDP_IFC.1/Import Note: FTP_ITC.1 or FTP_TRP.1 is not satisfied.
FPT_EMS_EXT.1	No dependency	NA
FPT_FLS.1/SB	No dependency	NA
FPT_FLS.1/SG	No dependency	NA
FPT_FLS.1/UD	No dependency	NA
FPT_PHP.3	No dependency	NA
FPT_RPL.1	No dependency	NA
FPT_TST.1	No dependency	NA
FPT_TUD_EXT.1	FCS_COP.1	Note: The ST author selects SFR.

The dependency FCS_COP.1 of FCS_STG_EXT.2, FCS_STG_EXT.3, FDP_MFW_EXT.1, and FPT_TUD_EXT.1 are to be filled in by the ST author, selected from the selection-based requirements of the cryptographic functional requirements that are being used.

The dependency FMT_MSA.3 of FDP_IFF.1/API is not satisfied. FMT_MSA.3 defines the management of “a security attribute of information.” In this TOE, this attribute is an API attribute that is calculated each time an API is called and is not managed by the TOE. Thus, FMT_MSA.3 is not applicable.

The dependency FMT_MSA.3 of FDP_IFF.1/Import is not satisfied. FMT_MSA.3 defines the management of “a security attribute of information.” In this TOE, this attribute is calculated outside of the TOE and is not managed by the TOE. Thus, FMT_MSA.3 is not applicable.

The dependency FTP_ITC.1 or FTP_TRP.1 of FDP_UIT.1 is not satisfied. The user data encrypted and MAC granted outside the TOE is imported, so the trusted path/trusted channel is not applied.

5.3.2 Security assurance requirements rationale

Security assurance requirements applied for the TOE are EAL1 + ASE_SPD.1 + ADV_ARC.1 + ADV_FSP.2 + ADV_TDS.1 + ALC_FLR.1 + AVA_VAN.2 and AVA_SCU_EXT.1. Extended security functional requirement AVA_SCU_EXT.1 is defined in Chapter 6.

Considering the value of the assets handled by the TOE and the operating environment in which the TOE is used, the decision to select these SARs was derived from being able to withstand the attacks of an attacker with basic attack potential (i.e., AVA_VAN.1 or AVA_VAN.2). Next, by considering the characteristics of the TOE where it is used for inexpensive devices, the level required for developers should be needed to obtain appropriate assurance while reducing the cost and time required for an evaluation and a certification.

However, the higher AVA_VAN.2 is selected because it can withstand the attacks of an attacker with basic attack potential (i.e., AVA_VAN.1 or AVA_VAN.2) and because of its dependencies ADV_ARC.1, ADV_FSP.2, and ADV_TDS.1 are selected. In addition to searching for known vulnerabilities, a meaningful

assurance of EAL1 or higher is obtained by performing a vulnerability analysis on the basis of the provided evaluation documentation, and independent testing based on the TOE specification.

In addition, it is impossible not to use configuration management, mass production, and logistics systems in recent semiconductor and embedded systems development, and from this perspective, the ALC class is the minimum to be included in EAL1. ALC_FLR.1 has also been added, and it is hoped that the TOE will be maintained in the future by tracking and correcting security flaws found by the developers. ASE_SPD.1 has been added to help the ST readers understand the use cases envisioned by the PP, threats, and assumptions associated with them.

AVA_SCU_EXT.1 Vulnerability survey of the SCU

The TOE is a hardware product that controls the cryptographic engine and hardware gate through the software gate. Although it is a microcontroller, this TOE does not handle high-value assets like microcontrollers for smart cards, and it is assumed that the asset value handled is low. Therefore, it is assumed that the attacker assumed in this TOE has a lower attack potential than one who attacks the smart card microcontroller.

In the evaluation by the Common Criteria, it is mandatory to apply [AAPS] to the vulnerability assessment of smart cards and similar devices. For this reason, a new extended assurance component AVA_SCU_EXT has been newly defined on the basis of [PPTEE], and the attack should be noted by the SCU developer and TOE developer who use the SCU, and an attack potential calculation table for hardware attacks based on the SOG-IS support document [AAPS] are defined. See Chapter 9 for AVA_SCU_EXT.

Table 5-3 shows the dependencies satisfied in the SARs of this TOE. As shown, all dependencies of the selected SARs are satisfied.

Table 5-3 Security assurance requirements

Requirements	CC dependencies	Satisfied dependencies
ASE_CCL.1	ASE_ECD.1, ASE_INT.1, ASE_REQ.1	ASE_ECD.1, ASE_INT.1, ASE_REQ.1
ASE_ECD.1	No dependency	NA
ASE_INT.1	No dependency	NA
ASE_OBJ.1	No dependency	NA
ASE_REQ.1	ASE_ECD.1	ASE_ECD.1
ASE_SPD.1	No dependency	NA
ASE_TSS.1	ASE_INT.1, ASE_REQ.1, ADV_FSP.1	ADV_FSP.2
ADV_ARC.1	ADV_FSP.1, ADV_TDS.1	ADV_FSP.2, ADV_TDS.1
ADV_FSP.2	ADV_TDS.1	ADV_TDS.1
ADV_TDS.1	ADV_FSP.2	ADV_FSP.2
AGD_OPE.1	ADV_FSP.1	ADV_FSP.2
AGD_PRE.1	No dependency	NA
ALC_CMC.1	ALC_CMS.1	ALC_CMS.1
ALC_CMS.1	No dependency	NA
ALC_FLR.1	No dependency	NA
ATE_IND.1	ADV_FSP.1, AGD_OPE.1, AGD_PRE.1	ADV_FSP.2, AGD_OPE.1, AGD_PRE.1

Requirements	CC dependencies	Satisfied dependencies
AVA_VAN.2	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1
AVA_SCU_EXT.1	AVA_VAN.1	AVA_VAN.2

6 Appendix: Extended component definitions

6.1 Extended security functional components

The extended components defined in this PP and the families that include them are shown in the following. These consisted of the [CC2] families and components as models and referenced as [PP0096] and [CPPFDE].

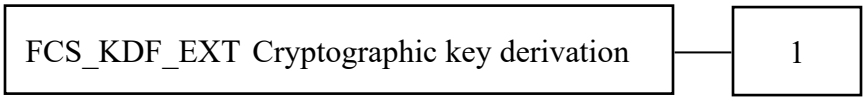
6.1.1 FCS_KDF_EXT Cryptographic key derivation

The creation of this family is necessary because [CC2] does not provide SFRs for derivation intermediate keys from the submask.

Family Behavior

This family specifies the means by which an intermediate key is derived from a specified set of submasks.

Component Leveling



FCS_KDF_EXT.1 Cryptographic Key Derivation requires the TSF to derive intermediate keys from submasks using the specified hash functions.

Management: FCS_KDF_EXT.1

No specific management functions are identified.

Audit: FCS_KDF_EXT.1

There are no auditable events foreseen.

FCS_KDF_EXT.1 Cryptographic Key Derivation

Hierarchical to:	No other components
Dependencies:	FCS_CKM.4 Cryptographic Key Destruction
	FCS_COP.1/MAC Cryptographic operation (Message authentication)
	FCS_RBG_EXT.1 Random bit generation

FCS_KDF_EXT.1.1 The TSF shall accept [selection: *an RNG generated submask as specified in FCS_RBG_EXT.1, imported submask*] to derive an intermediate key, as defined in [selection: *NIST SP 800-108 [selection: KDF in Counter Mode, KDF in Feedback Mode, KDF in Double-Pipeline Iteration Mode], NIST SP 800-132*] using the keyed-hash functions specified in FCS_COP.1/MAC, such that the output is at least of equivalent security strength (in number of bits) to the HGK.

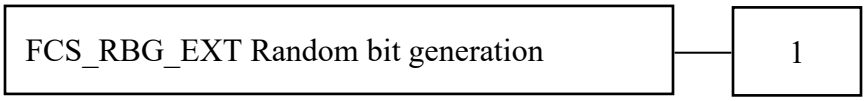
6.1.2 FCS_RBG_EXT Random bit generation

The creation of this family is necessary because [CC2] does not provide SFRs for random bit generation.

Family Behavior

Components in this family address the requirements for random bit/number generation. This is a new family defined for the FCS class.

Component Leveling



FCS_RBG_EXT.1 Random Bit Generation requires random bit generation to be performed in accordance with selected standards and seeded by an entropy source.

Management: FCS_RBG_EXT.1

No specific management functions are identified.

Audit: FCS_RBG_EXT.1

There are no auditable events foreseen.

FCS_RBG_EXT.1 Random Bit Generation

Hierarchical to: No other components

Dependencies: No dependencies

FCS_RBG_EXT.1.1 The TSF shall perform all deterministic random bit generation services in accordance with *ISO/IEC 18031:2011,* using [selection: *Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES)*].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from [selection:

- *[assignment: number of software-based sources] software-based noise source(s),*
- *[assignment: number of hardware-based sources] hardware-based noise source(s)]*

with a minimum of [selection: *128 bits, 192 bits, 256 bits*] of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 "Security Strength Table for Hash Functions," of the keys and hashes that it will generate.

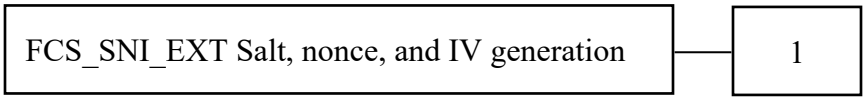
6.1.3 FCS_SNI_EXT Salt, Nonce, and Initialisation Vector Generation

The creation of this family is necessary because [CC2] does not provide SFRs for the generation of the salt, nonce, and IV.

Family Behavior

This family ensures that salts, nonces, and IVs are well formed.

Component Leveling



FCS_SNI_EXT.1 Cryptographic Operation (Salt, Nonce, and Initialisation Vector Generation) requires the generation of salts, nonces, and IVs to be used by the cryptographic components of the TOE to be performed in the specified manner.

Management: FCS_SNI_EXT.1

No specific management functions are identified.

Audit: FCS_SNI_EXT.1

There are no auditable events foreseen.

FCS_SNI_EXT.1 Salt, Nonce, and Initialisation Vector Generation

Hierarchical to: No other components

Dependencies: FCS_RBG_EXT.1 Random Bit Generation

FCS_SNI_EXT.1.1 The TSF shall use [selection: *no salts, salts that are generated by a [selection: DRBG as specified in FCS_RBG_EXT.1, submask provided by outside of the TOE at manufacturing]*].

FCS_SNI_EXT.1.2 The TSF shall use [selection: *no nonce, a unique nonce with a minimum size of [assignment: length of nonce] bits*].

FCS_SNI_EXT.1.3 The TSF shall create IVs in the following manner [selection:

- *CBC: IVs shall be non-repeating;*
- *CCM: Nonce shall be non-repeating;*
- *XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer;*
- *GCM: IV shall be non-repeating. The number of invocations of GCM shall not exceed 2^{32} for a given secret key*].

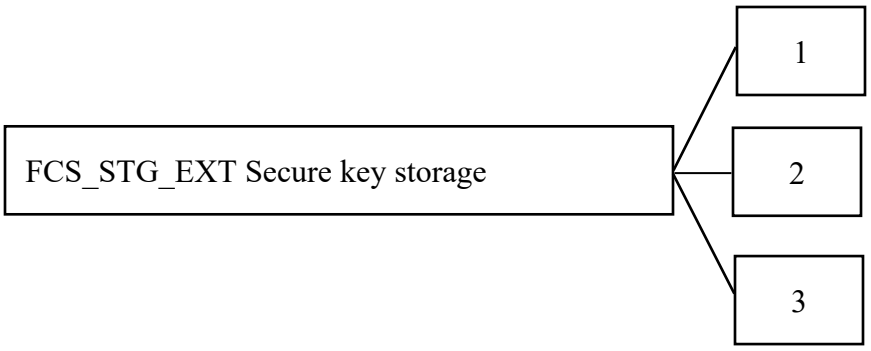
6.1.4 FCS_STG_EXT.1 Secure Key Storage

The creation of this family is necessary because [CC2] does not provide SFRs to store cryptographic keys.

Family Behavior

This family provides specifications for managing secure key storage outside of the SCU but within the TOE.

Component Leveling



FCS_STG_EXT.1 Secure key storage requires that the TSF maintain key storage and specify the characteristics of the storage.

FCS_STG_EXT.2 Key confidentiality protection requires that the TSF protects the confidentiality of the data stored in accordance with the specified method.

FCS_STG_EXT.3 Key integrity protection requires that the TSF maintains the integrity of the data stored in accordance with the specified method.

Management: FCS_STG_EXT.1、FCS_STG_EXT.2、FCS_STG_EXT.3

No specific management functions are identified.

Audit: FCS_STG_EXT.1、FCS_STG_EXT.2、FCS_STG_EXT.3

There are no auditable events foreseen.

FCS_STG_EXT.1	Secure key storage
Hierarchical to:	No other components
Dependencies:	No dependencies
FCS_STG_EXT.1.1	The TSF shall provide [selection: *immutable hardware-based, mutable hardware-based, software-based*] secure key storage for asymmetric private keys and [selection: *symmetric keys, persistent secrets, no other keys*].
FCS_STG_EXT.1.2	The TSF shall support the capability of importing keys/secrets into the secure key storage upon request of [selection: *the authorized subject*].
FCS_STG_EXT.1.3	The TSF shall be capable of destroying keys/secrets in the secure key storage upon request of [selection: *the authorized subject*].
FCS_STG_EXT.2	Key confidentiality protection
Hierarchical to:	No other components
Dependencies:	FCS_COP.1
FCS_STG_EXT.2.1	The TSF shall encrypt [assignment: *key data*] using the following method: [assignment: *method as specified in FCS_COP.1*.
FCS_STG_EXT.3	Key integrity protection

Hierarchical to: No other components

Dependencies: FCS_COP.1

FCS_STG_EXT.3.1 The TSF shall protect the integrity of [assignment: *key data*] by using [assignment: *method for integrity protection*].

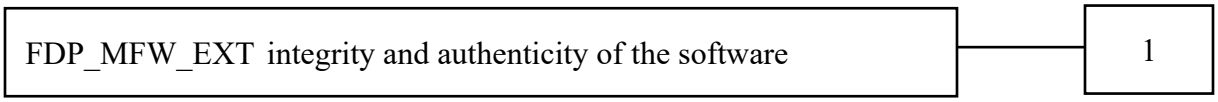
6.1.5 FDP_MFW_EXT.1 Integrity and authenticity of the software

The [CC2] requirements for protecting user data are broadly applicable, but this requirement is necessary because the specific verification of integrity and authenticity of the software is not defined.

Family Behavior

This family addresses the requirements of integrity and authenticity of the software for secure boot and trusted update.

Component Leveling



FDP_MFW_EXT.1 The software integrity and authenticity requires that the TSF verifyeither the integrity or authenticity of the software, or both.

Management: FDP_MFW_EXT.1

No specific management functions are identified.

Audit: FDP_MFW_EXT.1

There are no auditable events foreseen.

FDP_MFW_EXT.1 The software integrity and authenticity

Hierarchical to: No other components.

Dependencies: FCS_COP.1

FDP_MFW_EXT.1.1 The TSF shall have the ability to verify [selection: *integrity, authenticity*] of the software.

FDP_MFW_EXT.1.2 The TSF shall provide the capability to generate evidence of [selection: *integrity, authenticity*] of the software.

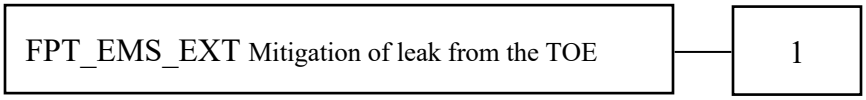
6.1.6 FPT_EMS_EXT Mitigation of leak from the TOE

Examples of exploiting leakage, such as TOE power consumption or electromagnetic radiation, include simple power analysis (SPA), differential power analysis (DPA), and timing attacks. The creation of this family is necessary because [CC2] does not define SFRs to mitigate unnecessary leaks.

Family Behavior

This family addresses the requirements for mitigation of unnecessary leaks.

Component Leveling



FPT_EMS_EXT.1 Mitigation of leak from the TOE requires mitigation of leakage that leads to the exposure of the TSF data and user data.

Management: FPT_EMS_EXT.1

No specific management functions are identified.

Audit: FPT_EMS_EXT.1

There are no auditable events foreseen.

FPT_EMS_EXT.1 Mitigation of leak from the TOE

Hierarchical to: No other components

Dependencies: No dependencies

FPT_EMS_EXT.1.1 The TOE shall not leak [*assignment: list of type of emissions*] over its [*assignment: list of attack surface*] in such an amount that these emissions enable access to [*assignment: list of type of TSF data*] and [*assignment: list of type of user data*].

6.1.7 FPT_TUD_EXT.1 Trusted update

The creation of this family is necessary because [CC2] does not define SFRs to apply update the software.

Family Behavior

This family addresses the requirements for updating the software.

Component Leveling

FPT_TUD_EXT.1 Trusted Update requires the capability to update the software, including the ability to verify the updates prior to installation.

Management: FPT_TUD_EXT.1

No specific management functions are identified.

Audit: FPT_TUD_EXT.1

There are no auditable events foreseen.

FPT_TUD_EXT.1 Trusted update

Hierarchical to: No other components

Dependencies: FCS_COP.1

FPT_TUD_EXT.1.1 The TSF shall provide [*assignment: list of subjects*] the ability to query the current version of the software.

FPT_TUD_EXT.1.2 The TSF shall provide [*assignment: list of subjects*] the ability to initiate updates to the software.

FPT_TUD_EXT.1.3 The TSF shall verify updates to the software using a [*selection: digital signature, MAC, [assignment: other method]*] by the manufacturer prior to installing those updates.

6.2 Extended security assurance component

6.2.1 AVA_SCU_EXT Vulnerability assessment of the SCU

Objectives

SCU vulnerability analysis is an assessment that determines whether a potential vulnerability enables an attacker to violate the SFRs, using the attack techniques identified in [AAPS]. This component is needed to define the attacks carried out by an attacker with basic attack potential that TOE developers incorporating SCUs should be aware of and to define a calculation table of attack potential of hardware attacks with reference to the SOG-IS support document [AAPS].

A vulnerability survey based on the attack method described in [AAPS] is performed by an evaluator to identify potential vulnerabilities that an attacker may exploit. The evaluator carries out penetration tests to ensure that potential vulnerabilities cannot be exploited in the TOE operational environment. The evaluator performs penetration tests assuming basic attack potential.

Component leveling

This family contains only one component.

AVA_SCU_EXT.1 Vulnerability survey of the SCU

Dependencies: AVA_VAN.1

Developer action elements:

AVA_SCU_EXT.1.1D The developer shall provide the TOE for testing.

Content and presentation elements:

AVA_SCU_EXT.1.1C The TOE shall be suitable for testing.

Evaluator action elements:

AVA_SCU_EXT.1.1E The evaluator *shall devise* penetration tests correspond to basic attack potential on the basis of SCU characteristics and *shall conduct* penetration testing.

6.2.2 Vulnerability survey of the SCU (Extended – AVA_SCU_EXT)

6.2.2.1 Evaluation of sub-activity (AVA_SCU_EXT.1)

6.2.2.1.1 Objective

The SCU Vulnerability Survey is conducted as part of the CEM AVA_VAN.1 evaluation activity and aims to refine the AVA_VAN.1 CEM work unit to identify vulnerabilities specific to the microcontroller equipped with the SCU.

6.2.2.1.2 Inputs

The evaluation evidence for this sub-activity is:

a) The TOE suitable for testing.

Other input for this sub-activity is:

a) Current information on potential vulnerabilities ([AMSS]).

6.2.2.1.3 Action AVA_SCU_EXT.1.1E

AVA_SCU_EXT.1.1C The TOE shall be suitable for testing.

AVA_SCU_EXT.1-1 The evaluator shall devise penetration tests correspond to basic attack potential on the basis of Chapter 9 and shall conduct penetration testing.

Chapter 9 provides a scale for calculating the attack potential required for an attacker to accomplish an attack. It also introduces the concept of an attack path consisting of one or more attack steps. To recognize a vulnerability, it is necessary to perform analysis and testing at each attack step of the attack path.

7 Appendix: Selection-based security functional requirements

The following are the SFRs to be selected to match the contents of the SFRs.

7.1 Cryptographic support

FCS_COP.1/SKC Cryptographic operation (symmetric key cryptography)

Hierarchical to: No other components

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation], FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/SKC The TSF shall perform [**selection: encryption, decryption**] in accordance with a specified cryptographic algorithm [**selection: cryptographic algorithm(s) from the following table**] and cryptographic key sizes [**selection: key size(s) from the following table**] that meet the following: [**selection: standard(s) from the following table**].

Table 7-1 Symmetric Key Cryptography

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
AES-CCM	AES in CCM mode with unpredictable, non-repeating nonce, minimum size of 64 bits	[selection: 128, 192, 256] bits	ISO/IEC 18033-3 (AES) ISO/IEC 19772, sec. 8 (CCM) NIST SP800-38C (CCM)
AES-GCM	AES in GCM mode with non-repeating IVs; the IV length must be equal to 96 bits; the deterministic IV construction method [SP800-38D, Section 8.2.1] must be used; the MAC length must be one of the values 96, 104, 112, 120, and 128 bits	[selection: 128, 192, 256] bits	ISO/IEC 18033-3 (AES) ISO/IEC 19772, sec.11 (GCM) NIST SP800-38D (GCM)
AES-CBC	AES in CBC mode with non-repeating and unpredictable IVs	[selection: 128, 192, 256] bits	ISO/IEC 18033-3 (AES) ISO/IEC 10116 (CBC) NIST SP800-38A (CBC)
AES-XTS	AES in XTS mode with unique [selection: consecutive non-negative integers starting at an arbitrary non-negative integer, data unit sequence numbers] tweak values	[selection: 128, 192, 256] bits	ISO/IEC 18033-3 (AES) [selection: IEEE 1619, NIST SP800-38E] (XTS)
AES-KWP	KWP based on AES	[selection: 128, 256] bits	ISO/IEC 18033-3 (AES), NIST SP 800-38F, sec. 6.3 (KWP) ISO/IEC 19772, clause 7 (key wrap)
AES-KW	KW based on AES	[selection: 128, 256] bits	ISO/IEC 18033-3 (AES), NIST SP 800-38F, sec. 6.2 (KW) ISO/IEC 19772, clause 7 (key wrap)

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
CAM-CBC	Camellia in CBC mode with non-repeating and unpredictable IVs	[selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (Camellia) ISO/IEC 10116 (CBC)
CAM-CCM	Camellia in CCM mode with unpredictable, non-repeating nonce, minimum size of 64 bits	[selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (Camellia) ISO/IEC 19772, sec. 8 (CCM) SP800-38C
CAM-GCM	Camellia in GCM mode with non-repeating IVs; the IV length must be equal to 96 bits; the deterministic IV construction method [SP800-38D, Section 8.2.1] must be used; the MAC length must be one of the values 96, 104, 112, 120, and 128 bits	[selection: 128 bits, 256 bits]	ISO/IEC 18033-3 (Camellia) ISO/IEC 19772, sec.11 (GCM) NIST SP800-38D
CAM-XTS	Camellia in XTS mode with unique [selection: consecutive non-negative integers starting at an arbitrary non-negative integer, data unit sequence numbers] tweak values	[selection: 256 bits, 512 bits]	ISO/IEC 18033-3 (Camellia) [selection: IEEE 1619, SP800-38E] (XTS)
ChaCha-Poly	ChaCha20-Poly1305 with unpredictable, non-repeating nonce, minimum size of 96 bits	256 bits	RFC 8439

Application Note 16

For the Camellia and ChaCha20-Poly1305 cryptographic algorithms, developers must consult with certification bodies before selecting them, as cryptographic algorithm testing and side-channel information measurement methods are still under consideration.

FCS_COP.1/KeyEnc Cryptographic operation (key encryption)

Hierarchical to: No other components

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation], FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/KeyEnc The TSF shall perform [*key encryption and decryption*] in accordance with a specified cryptographic algorithm **[selection: cryptographic algorithm from the following table]** and the cryptographic key size **[selection: key size from the following table]** that meet the following: **[selection: list of standards specified in FCS_COP.1/SKC]**.

Table 7-2 Key encryption

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
SE1	[selection: AES-CCM, AES-GCM, AES-CBC]	[selection: 128, 192, 256] bits	Select a list of standard from FCS_COP.1/SKC

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
SE2	[selection AES-KWP, AES-KW, CAM-CBC, CAM-CCM, CAM-GCM]	[selection: 128, 256] bits	that corresponds to an algorithm in the left column.

FCS_COP.1/Hash Cryptographic operation (hashing)

Hierarchical to: No other components

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation], FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/Hash The TSF shall perform [*hashing*] in accordance with a specified cryptographic algorithm **[selection: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE256]** that meets the following: **[selection: ISO/IEC 10118-3:2018, FIPS 180-4, FIPS202]**.

Application Note 17

The hash selection should be consistent with the overall strength of the algorithm used for signature generation. For example, the TOE should choose SHA-256 for ECC with P-256, SHA-384 for ECC with P-384, and SHA-512 for ECC with P-521.

SHA-1 may be used for the following applications: generating and verifying hash-based message authentication codes (HMACs), key derivation functions (KDFs), and random bit/number generation.

FCS_COP.1/MAC Cryptographic operation (MAC)

Hierarchical to: No other components

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation], FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/MAC The TSF shall perform cryptographic [*message authentication*] in accordance with a specified cryptographic algorithm [**selection: HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, CMAC-AES-128, CMAC-AES-256**] and cryptographic key sizes [***assignment: key size (in bits)***] **used in [selection: HMAC, AES]** that meet the following: [**selection: ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2," NIST SP 800-38B**].

Application Note 18

If one or more HMAC algorithms are selected, the ST author selects "HMAC" in the second selection and "ISO/IEC 9797-2:2011, Section 7 'MAC Algorithm 2'" in the third selection. For the assignment, the key size [k] falls into a range between L1 and L2 (defined in ISO/IEC 10118 for the appropriate hash function). For example, for SHA-256, L1=512 and L2 =256, where L2 ≤ k ≤ L1.

If one or more CMAC algorithms are selected, the ST author selects "AES" in the second selection and "NIST SP 800-38B" in the third selection. For the assignment, the key size will fall into a range between 128 and 256.

FCS_COP.1/SigVer Cryptographic Operation (Digital Signature Verification)

Hierarchical to: No other components

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation], FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/SigVer The TSF shall perform [*digital signature verification for authenticity*] in accordance with a specified cryptographic algorithm [**selection: cryptographic algorithm from the following table**] and cryptographic key sizes [**selection: key size(s) from the following table**] that meet the following: [**selection: list of standards from the following table**].

Table 7-3 Signature verification

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
ECDSA	ECDSA on [selection: brainpoolP256r1, brainpoolP384r1, brainpoolP512r1, NIST P-256, NIST P-384, NIST P-521] using [selection: SHA-256, SHA-512]	[selection: 256 bits, 384 bits, 512 bits]	[selection: ISO/IEC14888-3; FIPS186-4 (Section 6)] [ECDSA] RFC5639 (Section 3) [Brainpool Curves] FIPS186-4 (Appendix D.1.2) [NIST Curves] [selection: ISO/IEC10118-3, (Section 10, 11); FIPS180-4, (Section 6)] [SHA]
EdDSA	EdDSA [selection: Ed25519, Ed448] using [selection: SHA-512, SHAKE256]	[selection: 256 bits, 456 bits]	RFC8032 [EdDSA] [selection: ISO/IEC10118-3, (Section 10, 11); FIPS180-4, (Section 6), FIPS202 (section 6)] [SHA]

FCS_KDF_EXT.1 Key derivation functions

Hierarchical to: No other components

Dependencies: FCS_CKM.4 Cryptographic key destruction, FCS_COP.1/MAC Cryptographic operation (message authentication), FCS_RBG_EXT.1 Random bit generation

FCS_KDF_EXT.1.1 The TSF shall accept [selection: *an RNG generated submask as specified in FCS_RBG_EXT.1, imported submask]* to derive an intermediate key, as defined in [*NIST SP 800-108 [selection: KDF in Counter Mode, KDF in Feedback Mode, KDF, in Double-Pipeline Iteration Mode]*] using the keyed-hash functions specified in FCS_COP.1/MAC, such that the output is at least of equivalent security strength (in number of bits) to the HGK.

Application Note 19

The documentation of the product's encryption key management should be detailed enough that, after reading, the evaluator will thoroughly understand the product's key management and how it meets the requirements to ensure the keys are adequately protected.

FCS_RBG_EXT.1 Random bit generation

Hierarchical to: No other components

Dependencies: No dependencies

FCS_RBG_EXT.1.1 The TSF shall perform all deterministic random bit generation services in accordance with ISO/IEC 18031:2011 using [selection: Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES)].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by at least one entropy source in accordance with NIST SP 800-90B that accumulates entropy from [[*assignment: number of hardware-based sources*] *hardware-based noise source*] with a minimum of [selection: 128 bits, 192 bits, 256 bits] of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 “Security Strength Table for Hash Functions,” of the keys that it will generate.

Application Note 20

NIST Special Pub 800-90B, Appendix C describes the minimum entropy measurement that products should use immediately. The RBG shall generate sufficient entropy. ISO/IEC 18031:2011 contains four different methods of generating random bits. Each of these in turn depends on the underlying cryptographic primitives (hash functions/ciphers). This PP allows SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 for Hash_DRBG or HMAC_DRBG and only AES-based implementations for CTR_DRBG to be selected.

FCS_SNI_EXT.1 Salt, Nonce, and Initialisation Vector Generation

Hierarchical to: No other components

Dependencies: FCS_RBG_EXT.1 Random bit generation

FCS_SNI_EXT.1.1 The TSF shall use [selection: no salts, salts that are generated by a [selection: DRBG as specified in FCS_RBG_EXT.1, submask provided by outside of the TOE at manufacturing].

FCS_SNI_EXT.1.2 The TSF shall use [selection: no nonce, a unique nonce with a minimum size of [*96*] bits].

FCS_SNI_EXT.1.3 The TSF shall create IVs in the following manner [selection:

- CBC: IVs shall be non-repeating;
- CCM: Nonce shall be non-repeating;
- XTS: No IV. Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer;
- GCM: IV shall be non-repeating. The number of invocations of GCM shall not exceed 2^{32} for a given secret key].

8 Appendix: Option-based security functional requirements

The following optional SFRs can be added.

8.1 Cryptographic support

FCS_CKM.1/AK Cryptographic key generation (asymmetric keys)

Hierarchical to: No other components

Dependencies: [FCS_CKM.2 Cryptographic key distribution, or FCS_COP.1 Cryptographic operation], FCS_CKM.4 Cryptographic key destruction

FCS_CKM.1.1/AK The TSF shall generate **asymmetric** cryptographic **keys [selection: key name from the following table]** in accordance with a specified cryptographic key generation algorithm **[selection: cryptographic key generation algorithm from the following table]** and specified cryptographic key sizes **[selection: key size(s) from the following table]** that meet the following: **[selection: list of standards from the following table].**

Table 8-1 List of asymmetric keys

Key Name	Cryptographic Key Generation Algorithm	Key Sizes	List of Standards
ECC-N	FIPS PUB 186-4 (Section B.4)	[selection: 256 (P-256), 384 (P-384), 512 (P-521)]	FIPS PUB 186-4 (Section B.4 & D.1.2)
ECC-B	FIPS PUB 186-4 (Section B.4)	[selection: 256 (brainpoolP256r1), 384 (brainpoolP384r1), 512 (brainpoolP512r1)]	RFC5639 (Section 3) [Brainpool Curves] FIPS PUB 186-4 (Section B.4)

FCS_CKM.1/SK Cryptographic key generation (symmetric encryption key)

Hierarchical to: No other components

Dependencies: [FCS_CKM.2 Cryptographic key distribution, or FCS_COP.1 Cryptographic operation], FCS_CKM.4 Cryptographic key destruction

FCS_CKM.1.1/SK The TSF shall generate **symmetric** cryptographic keys **[selection: key name from the following table]** in accordance with a specified cryptographic key generation algorithm **[selection: cryptographic key generation algorithm from the following table]** and specified cryptographic key sizes **[selection: key size(s) from the following table]** that meet the following: **[selection: list of standards from the following table]**.

Table 8-2 Symmetric key generation

Key Name	Cryptographic Key Generation Algorithm	Key Sizes	List of Standards
RSK	Direct Generation from a Random Bit Generator as specified in FCS_RBG_EXT.1	[selection: 128 bit, 256 bit, 512 bit]	NIST SP 800-133 (Section 7.1) with ISO/IEC 18031 as an approved RBG in addition to those in NIST SP 800-133 (Section 5)

Key Name	Cryptographic Key Generation Algorithm	Key Sizes	List of Standards
DSK	[selection: Key Derivation Functions as specified in FCS_COP.1/KDF]	[selection: 128 bit, 256 bit, 512 bit]	NIS TSP 800-108

Application Note 21

The key size selection of 512 bits is for the case of XTS-AES using AES-256. In the case of XTS-AES for both AES-128 and AES-256, the developer should ensure that the full key is generated using direct generation from the RBG as in the NIST SP800-133 section.

The ST author selects a Random Symmetric Key (RSK) if the TOE supports generating keys directly from the output of the RBG without further conditioning. A Derived Symmetric Key (DSK) should be selected if the TOE supports Key Derivation Function (KDF)s that are usually seeded from the RBG and then further conditioned to the appropriate key size.

FCS_CKM.4 Cryptographic key destruction

Hierarchical to: No other components

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4.1 The TSF shall destroy cryptographic keys **and keying material** in accordance with a specified cryptographic key destruction **method [**

For volatile memory, the destruction shall be executed by a [selection:

a. **single overwrite consisting of [selection:**

i. **a pseudo-random pattern using the TSF's RBG,**

ii. **zeroes,**

iii. **ones,**

iv. **a new value of a key,**

v. **[*assignment: some value that does not contain any confidential information*]],**

b. **removal of power to the memory,**

c. **removal of all references to the key directly followed by a request for garbage collection];**

] that meets the above: [*no standard*].

Application Note 22

In the case of volatile memory, the selection "destruction of reference to the key directly followed by a request for garbage collection" is used in a situation where the TSF cannot address the specific physical memory locations holding the data to be erased and therefore relies on addressing logical addresses (which frees the relevant physical addresses holding the old data) to ensure that the data in the physical addresses is no longer available for reading (i.e., the "garbage collection" referred to in the SFR text). The term "removal of power to the memory " is assumed to turn off the main power supply of the TOE and does not assume a means to cut off the power supply to the volatile memory by a circuit like a switch.

FCS_COP.1/SigGen Cryptographic operation (Digital signature generation)

Hierarchical to: No other components

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation], FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/SigGen The TSF shall perform [*digital signature generation*] in accordance with a specified cryptographic algorithm [**selection: cryptographic algorithm(s) from the following table**] and cryptographic key sizes [**selection: key size(s) from the following table**] that meet the following: [**selection: standard(s) from the following table**].

Table 8-3 Digital signature generation

Identifier	Cryptographic Algorithm	Key Sizes	List of Standards
ECDSA	ECDSA on [selection: brainpoolP256r1, brainpoolP384r1, brainpoolP512r1, NIST P-256, NIST P-384, NIST P-521] using [selection: SHA-256, SHA-512]	[selection: 256 bits, 384 bits, 512 bits, 521 bits]	[selection: ISO/IEC14888-3; FIPS186-4 (Section 6)] [ECDSA] RFC5639 (Section 3) [Brainpool Curves] FIPS186-4 (Appendix D.1.2) [NIST Curves] [selection: ISO/IEC10118-3, (Section 10, 11); FIPS180-4, (Section 6)] [SHA]
EdDSA	EdDSA [selection: Ed25519, Ed448] using [selection: SHA-512, SHAKE256]	[selection: 256 bits, 456 bits]	RFC8032 [EdDSA] [selection: ISO/IEC10118-3, (Section 10, 11); FIPS180-4, (Section 6), FIPS202 (section 6)] [SHA]

9 Appendix: AVA_SCU_EXT - Vulnerability survey of the SCU

This chapter introduces the methodology for evaluating the attack potential for the TOE equipped with the SCU, which has been developed by industry stakeholders (universities, SCU developers, integrators, evaluation labs, and certification bodies) within the SCU inside committee.

The goal of the methodology is to help evaluate the effort required to perform a successful attack on a given TOE. It provides the definition of the TOE equipped with the SCU-specific attack quotation table and guidance to calculate the potential required by an attacker to perform an attack through examples.

9.1 Identification of factors and rating attack potential

In the Common Criteria, there is no distinction between the identification and exploitation phases of an attack. However, within the smart card 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 that of "exploitation" (e.g., once a script is published on the Internet). Therefore, this distinction must be made when calculating the attack potential for the TOE equipped with the SCU 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.

9.1.1 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. See [AAPS] Chapter 4 for an idea of the attack identification and exploitation phases.

9.1.2 Elapsed time

"Elapsed time" is the time required to identify and exploit a vulnerability. This document follows [AAPS] 4.2 and is as shown in the following table. See [AAPS] 4.2 for details on the elapsed time.

Table 9-1 Elapsed time

	Identification	Exploitation
< one hour	0	0
< one day	1	3
< one week	2	4
< one month	3	6
> one month	5	8
> six month	6	10
Not practical	*	*

Elapsed time ranges for one day is about 8 hours, those for one week is about 40 hours, and those for one month is about 160 hours.

9.1.3 Expertise

Expertise is the technical expertise needed to identify and exploit vulnerabilities. This document follows [AAPS] 4.3 and is as shown in the following table. See [AAPS] 4.3 for details on the elapsed time.

Table 9-2 Expertise

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

9.1.4 Knowledge of TOE

Knowledge of TOE is the knowledge of TOE design and operation required to identify and exploit vulnerabilities. This document follows [AAPS] 4.4 and is as shown in the following table. See [AAPS] 4.4 for details on the elapsed time.

Table 9-3 Knowledge of TOE

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

9.1.5 Access to TOE

Access to TOE is the number of TOE required to identify and exploit a vulnerability. This document follows [AAPS] 4.5 and is as shown in the following table. See [AAPS] 4.5 for details on the elapsed time.

Table 9-4 Access to TOE

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

9.1.5.2 Rating the efficacy of TOE package

The TOE can select from a variety of packages. It is necessary to physically remove the package to attack the TOE, and the removal procedure is part of the attack path. Depending on how difficult it is to remove

the package, the rate will be as follows. This document follows [AAPS] 4.6 and is as shown in the following table. See [AAPS] 4.6 for details on the elapsed time.

Table 9-5 TOE package removal

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

9.1.6 Equipment

The equipment for attacking the TOE is divided into the following four categories and weighted. See [AAPS] 4.6 for a definition of equipment refinement.

Table 9-6 Equipment

	Identification	Exploitation
None	0	0
Standard	1	2
Specialised	3	4
Bespoke	5	6
Multiple bespoke	7	8

The purchase price is one of the indicators when refining standard, specialised, and bespoke equipment. According to [AAPS] 4.6.1, the details of the equipment in accordance with the purchase price are as follows.

Table 9-7 Equipment rating versus purchasing cost

Purchasing Cost	Equipment Rating
Up to 10 K€ (Up to 1.2M yen for 120 yen per 1 €)	Standard
Between 10 K€ and 200 K€ (1.2M yen to 24M yen)	Specialised
Over 200 K€ (Over 24M yen)	Bespoke

See [AAPS] 4.6.1 for specific examples that apply to each refined equipment.

9.1.7 Open Samples/Samples with known Secrets

The following definition summarizes [AAPS] 4.7. See [AAPS] 4.7 for details.

The term “open sample” refers to samples with the capability to download and/or run any kind of test software, and such samples may enable 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 environments by the vendor. However, the IC hardware shall not be changed.

“Samples with known secrets” refer to samples that can set secrets such as a PIN and key in the TOE by using functions that cannot be used in the normal operation of the TOE. If the vendor gives specific access to internal secrets, “samples with known secrets” can be considered.

Every functional interface or key necessary for the functional tests of the TOE provided by the vendor to the evaluation lab shall not be considered as an “open sample/sample with known secrets.”

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. These are forbidden to be used in the exploitation phase.

Table 9-8 Samples with known secrets

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

Table 9-9 Open sample

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

9.1.8 Calculation of attack potential

Create a table as follows and add the aforementioned factors required for an attack to obtain the total points.

Table 9-10 Calculation table of attack potential

Factor	Identification	Exploitation
Elapsed Time		
Expertise		
Knowledge of TOE		
Access to TOE		
Equipment		
Open sample/Sample with known secret		
Subtotal		
Total		

Convert the total points to attack potential on the basis of the following table.

Table 9-11 Rating of vulnerabilities and TOE resistance

Range of values	TOE resistant to attackers with attack potential of:
0–15	No rating

Range of values	TOE resistant to attackers with attack potential of:
16–20	Basic
21–24	Enhanced-Basic
25–30	Moderate
31 and above	High

This TOE must be able to withstand the attacks of an attacker with basic attack potential. That is, the total points of factors required for an attack must reach 16 points. On the other hand, the TOE may assume an attack by an attacker with an attack potential of more than 20 points, but the balance between assets handled by the TOE and the development cost of security measures should be considered.

9.1.9 Attacker Profiles in the TOE

The use of the TOE does not assume high-value assets such as financial or personal information stored inside smart cards. Therefore, it is unlikely that a person with high expertise will attack the TOE using expensive equipment and acquire a low asset value. The attacker assumed is considered to be a pleasant criminal who wants to show off their technology. The pleasant criminal owns common equipment and is likely to be able to use the slightly more expensive specialised equipment found in universities and businesses.

Each factor has a relationship. Since it is unlikely that a layman will master specialised equipment, a combination of a "proficient" and specialised equipment is appropriate. It also removes a medium effort package, which may result in a smaller sample size if an attacker uses specialised equipment; however, when a layman uses standard equipment, it seems that many samples are used by trial and error. Similarly, a "proficient" may use specialised equipment to successfully perform a side-channel attack in a short time, or a layman may continue to attempt side-channel attacks for some time using standard equipment.

In this way, the profile of the attacker who has the basic attack potential assumed by this TOE is determined, and the elapsed time, expertise, knowledge of TOE, access to the TOE, and equipment required to attack the TOE can be assumed.

9.1.9.1 Elapsed time

Since TOE does not have a high-value asset, even if an attacker succeeds in one attack pass at the cost of more than four months, profits or show off fame is small. Therefore, it is assumed that an attacker with basic attack potential spends more than one month and less than four months to attack. Therefore, the elapsed time is one of "<1 hour," "<1 day," "<1 week," "<1 month," and ">1 month."

9.1.9.2 Expertise

Assume the expertise of an attacker with basic attack potential as follows.

1. Physical attack, Overcome sensors or filters

   The expertise required to remove the TOE package, capture memory content, and change circuit behavior by cutting or shortening wires is considered a "proficient." It is assumed that an "expert" and "multiple experts" are not involved in this attack. In addition, the tools used to remove packages required for physical attacks are precision polishing equipment, etching (plasma etching and chemical etching), etc., and some experience is required to master these, so a "layman" is not assumed.

2. Perturbation attack, Attack on RNG, Exploitation test features

   The expertise required for attacks that exposes the TOE to an environment that exceeds its operating specifications and causes it to malfunction, and apply energy to the TOE and cause malfunctions is a

"layman" and "proficient." It is possible for even a "layman" to apply a high-voltage pulse to the TOE, cause a momentary power failure, and raise a part of the circuit to a high temperature. The attacker who uses a laser workstation or an electromagnetic field imaging (EMFI)/Body Biasing Injection (BBI) workstation is considered a "proficient" due to the experience needed to master them.

3. Side-channel attacks

   The expertise required for side-channel attacks is a "layman" and "proficient." Since a measurement method by connecting a shunt resistor for power consumption acquisition and analysis programs for statistical analysis are widely disclosed on the Internet, even a "layman" can attempt such attacks. Measurement of radiated electromagnetic waves and precise acquisition waveform alignment require a "proficient" level skill. However, from the perspective of asset value, an attacker with more than a "proficient" level expertise is unexpected.

4. Software attacks

   The expertise required for software attacks is a "layman" and "proficient." The software gate API is not disclosed to the public, but free tools are available on the Internet for fuzzing to search for application bugs from external interfaces, and even a "layman" can attempt fuzzing. Filtering the exploration parameters of fuzzing requires protocol specific expertise, but does not require an expert level, and is assumed to be a "proficient."

9.1.9.3 Knowledge of TOE

Knowledge of TOE of an attacker with basic attack potential is assumed to be “public.” In other words, information above the public level is protected by the TOE developer.

9.1.9.4 Access to TOE

Access to TOE of an attacker with basic attack potential is assumed as follows.

1. Physical attack, Overcome sensors or filters

   Trial and error of attack is required to remove the TOE package. Assuming that the attacker level is "proficient" and the attacker masters handling equipment classified as specialised such as Focused Ion Beam (FIB), the total points of equipment and expertise is 16. Since the upper limit of the basic attack potential is 20 points, the number of samples is less than 10. If the attacker is a layman, the number of failure samples will increase because the attacker will make trial and error attempts using standard equipment, but the number of samples will be less than 30 in relation to the elapsed time. On the basis of the aforementioned assumptions, it is assumed that the number of "access to TOE" is less than 30 at most. Therefore, it becomes either "<30 samples" or "<10 samples."

2. Perturbation attack, Attack on RNG, Exploitation test features

   Smart card ICs that can withstand an attacker with high attack potential may have a velocity counter. This is a mechanism that counts up when abnormal energy is applied and is detected by the sensor, and when a certain count threshold is exceeded, the operation of the IC is permanently stopped or confidential information is erased. In this case, a large number of samples are required to access due to trial-and-error attacks. For this TOE, it is assumed that the user wants to continue the operation rather than stopping it to protect the assets, so it is considered that only one TOE is required for access. Therefore, "<10 samples."

3. Side-channel attacks, Software attacks

   It is possible to construct an attack path with one sample, and it is also possible to abuse it using that attack path with one sample. Therefore, "<10 samples."

As for the effort for TOE package preparation, the “high preparation effort” is unexpected since it requires "multiple experts" and "bespoke" equipment, which deviates from the assumption of an attacker with basic attack potential. Therefore, it becomes either "low preparation effort" or "medium preparation effort."

9.1.9.5 Equipment

The equipment used by attackers with basic attack potential is "standard" and "specialised." The TOE does not deal with high-value assets. To expose low-value assets, it is not assumed that an attack will be carried out using "bespoke" equipment of about 24 million yen or more. It is assumed that standard equipment that can be easily purchased by university officials or pleasant criminals who want to show off their skills, and specialised equipment owned by universities and research institutes will be used.

9.1.9.6 Open sample/Samples with known secrets

“Open sample/Samples with known secret” is not assumed. According to [AAPS] paragraph 77, the functional interfaces and keys used for functional testing do not apply to open samples/samples with known secrets. Therefore, it becomes "none."

9.1.10 Ratings of attack potential in this TOE

The following is a rating table of the attack factors in this TOE.

Table 9-12 Calculation table for attack potential

Factor	Identification	Exploitation
Elapsed time		
< one hour	0	0
< one day	1	3
< one week	2	4
< one month	3	6
> one month	5	8
Expertise		
Layman	0	0
Proficient	2	2
Knowledge of TOE		
Public	0	0
Access to TOE		
<10 samples	0	0
<30 samples	1	2
The effort for TOE package preparation		
Low preparation effort	0	0
Medium preparation effort	1	2
Equipment		
None	0	0

Factor	Identification	Exploitation
Standard	1	2
Specialised	3	4
Open sample/Samples with known secrets		
None	0	NA

The following table is the result of combining the factors in the table of attack potential so that the basic attack potential is 16 to 20 points. Here, it is assumed that an attacker who can handle "specialised" equipment is a "proficient," and that the number of samples with "medium preparation effort" can be successfully removed with "less than 30 samples." In addition, "knowledge of TOE" is "public," and "open sample/samples with known secrets" is "none," which are both 0 points.

Table 9-13 Basic attack potential

Factor	Combination 1				Combination 2			
	Identification		Exploitation		Identification		Exploitation	
Elapsed time	> one month	5	< one month	6	< one week	3	< one day	3
Expertise	layman	0	layman	0	proficient	2	proficient	2
Access to TOE	< 30	1	< 10	2	< 30	1	< 10	0
Package removal	medium	1	medium	2	medium	1	medium	2
Equipment	standard	1	standard	2	specialised	3	specialised	4
Subtotal	8		12		10		11	
Total	20				21			

Factor	Combination 3				Combination 4			
	Identification		Exploitation		Identification		Exploitation	
Elapsed time	> one month	5	< one month	6	< one month	3	< one week	4
Expertise	layman	0	layman	0	proficient	2	proficient	2
Access to TOE	< 10	0	< 10	0	< 10	0	< 10	0
Package removal	low	0	low	2	low	0	low	0
Equipment	standard	1	standard	2	specialised	3	specialised	4
Subtotal	6		10		8		10	
Total	16				18			

Factor	Combination 5				Combination 6			
	Identification		Exploitation		Identification		Exploitation	
Elapsed time	> one month	5	< one month	6	< one week	2	NA	0
Expertise	proficient	2	layman	0	expert	5	NA	0
Access to TOE	< 10	0	< 10	0	< 10	0	NA	0
Package removal	low	0	low	2	medium	1	NA	0
Equipment	standard	1	standard	2	specialised	3	NA	0

Subtotal	8	10	11	0
Total	18		11	

9.2 Examples of attack

The following examples have been compiled by JHAS, which is a group of security experts representing the different actor groups involved in the development, production, security evaluation, and distribution of a smart card product (hardware vendors, card vendors, OS provider, evaluation labs, Certification Bodies, service providers).

The following example summarizes [AAPS] Chapter 5. See [AAPS] Chapter 5 for details. The details of the attacks are described in the private [AMSS]. Contact ICSS-JC to obtain [AMSS].

Examples of calculating the attack potential applicable to TOE is shown in the following. This was created for the reader's understanding.

9.2.1 Physical attack

Using various equipment used for semiconductor failure analysis, the attacker can remove the semiconductor package to access the inside of the semiconductor, or add wires to tamper with the circuit. By these processes, the internal signal can be exposed and the behavior can be modified. In addition, physical access is performed to read the bit value of the memory or forcibly set the bit value.

The main impacts are:

- Access to secret data such as cryptographic keys
- Disabling IC security features to make another attack easier such as fault injection or DFA
- Forcing internal signals (Forcibly set to 0 (L) or 1 (H))

Assumed attack path is as follows.

1. Bus probing

   To carry out this attack, the attacker may remove the package, remove the wiring layers and insulating film, grasp the circuit layout, and identify the bus from the peripheral circuits of the memory block. Processing by FIB is required to connect tungsten wires to the bus while the circuit is operating. For an 8-bit bus, processing is required 8 times. The attacker will pull out the wires from the bus and analyze the signal with a bus analyzer. This attack requires "specialised" equipment and an "expert," and the elapsed time for identification should exceed one month, so an attacker with basic attack potential will not succeed.

2. Restoring non-volatile memory content

   Since an atomic force microscope or an expensive SEM is required to get an image of FLASH memory, it exceeds basic attack potential. For read-only memory (ROM), the "expert" level is required for removing the package and wiring layer, taking pictures with a metallurgical microscope or a cheap SEM, stitch images together, and then perform a software analysis. This is combination 6 in Table 9-13, never store secrets such as keys or PINs in ROM.

9.2.2 Overcoming sensors and filters

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

The main impacts are:

- Contents of memory or registers may be corrupted
- Program flow may be changed
- Failures in operations may occur (e.g., CPU, cryptographic engine)
- Change of operating mode and/or parameters

Assumed attack path is as follows.

1. Deactivate voltage sensors

   To carry out this attack, the attacker may remove the package, remove the wiring layers and insulating film, grasp the circuit layout, and identify the bus from the peripheral circuits of the power supply block. Processing by FIB is required to cut or shorten wires to deactivate sensor while making the sensor circuit operating. This attack requires "specialised" equipment and an "expert," and the elapsed time for identification should exceed one month, so an attacker with basic attack potential will not succeed.

9.2.3 Perturbation attacks, Attack on RNG, Exploitation of test features

The perturbation attack is also called a fault injection attack. This is an attack that causes a malfunction that can be exploited in the TOE. There is a method of exposing the TOE to an abnormal operating environment as described in the previous section, and a method of disturbing the TOE by applying energy such as strong light, electromagnetic waves, and high-voltage pulse waves to a power source from the outside of the TOE.

Attack on RNG causes a malfunction by an injection fault into a RNG circuit.

Exploitation test of a test feature causes a malfunction in a secure boot program to transit a test mode by injection fault.

The main impacts are:

- Values may be corrupted during reading memory
- Contents of memory or registers may be corrupted. The behavior of the TOE may be changed by changing a condition register or a failure counter, and tampering with the contents of an instruction register or program counter
- Failures in the cryptographic engine to perform DFA
- Failures in the RNG to output low quality random bits

Assumed attack path is as follows.

1. Fault injection by laser, EMFI, and BBI

   To carry out this attack, the attacker may remove the package, expose the substrate of the SoC, irradiate and move a laser spot by a laser work station to cause a malfunction while the TOE is operating. Similarly, the BBI uses the BBI workstation to repeatedly contact, apply, and move a probe to the substrate. For EMI, the attacker may bring a small coil closer to the substrate, irradiate, and move

repeatedly. Assuming that trial and error is required to remove the package and workstations are equivalent to "specialised" equipment, countermeasures that exceed the rates equivalent to combinations 2 or 4 in Table 9-13 are required.

2. Fault injection by voltage glitch

   To carry out this attack, a function generator causes the voltage of the power supply to spike or momentarily stop, causing a malfunction. The function generators are "standard" equipment, and depending on their price, a layman can easily attempt the attack. Countermeasures that exceed the rates equivalent to combinations 3 or 5 in Table 9-13 are required.

9.2.4 Side-channel attacks

Side-channel attacks measure power consumption and radiated electromagnetic during cryptographic operations, and use unintended leaked information in algorithm implementation to infer confidential information such as PINs and encryption keys.

The main impacts are:

- Exposing secret information such as keys and PINs by statistical analysis.
- Estimate the behavior of the TOE such as write timing to EEPROM and PIN comparison.

Assumed attack path is as follows.

1. Analyze by power consumption waves

   In this attack, a shunt resistor is connected to the power supply terminal of the SoC, the current during encryption calculation execution is measured, and the encryption key is inferred by statistical analysis. It can be measured with a standard oscilloscope and analyzed with statistical analysis software available on the Internet. Countermeasures that exceed the rates equivalent to combinations 3 or 5 in Table 9-13 are required.

2. Analyze by radiated electromagnetic waves

   To carry out this attack, the attacker may remove the package, expose the substrate of the SoC, bring a small coil closer to the substrate, and measure radiated electromagnetic waves. Appropriate noise filters are required to remove noise. A high-end oscilloscope classified as "specialised" is required to acquire the leak in the noise and, a "proficient" level expertise is required for appropriate coil placement and noise removal at the identification phase. Countermeasures that exceed the rates equivalent to combinations 4 or 5 in Table 9-13 are required.

9.2.5 Software attacks

The software attack is an attack that exploits an overflow to expose data in the receive buffer, exposes command data by abusing the protocol re-send command, or exploits software bugs or Reconfigurable Functional Unit (RFU) implementations of protocols.

The main impacts are:

- Exposing secret data

Assumed attack path is as follows.

1. Analyzing protocol

The attacker may set data in the RFU part of the standard protocol and observe the behavior. Countermeasures that exceed the rates equivalent to combinations 3 or 5 in Table 9-13 are required since a layman can easily attempt the attack using "standard" equipment.

2. Analyzing by fuzzing

   By comprehensively creating possible values of commands and sending them to the TOE, the attack may observe the behavior of the TOE when data outside the protocol specifications is received. Since the behavior of the TOE becomes either "no response" or "reset", commands are repeatedly sent while observing the behavior using a special device. Countermeasures that exceed the rates equivalent to combinations 4 or 5 in Table 9-13 are required.

End of document.